A NEW COMPETITIVE RATIO FOR NETWORK APPLICATIONS WITH HARD PERFORMANCE GUARANTEE

A Dissertation

by

HAN DENG

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	I-Hong Hou
Committee Members,	P. R. Kumar
	Radu Stoleru
	Le Xie
Head of Department,	Miroslav M. Begovic

August 2017

Major Subject: Computer Engineering

ABSTRACT

Online algorithms are used to solve the problems which need to make decisions without future knowledge. Competitive ratio is used to evaluate the performance of an online algorithm. This ratio is the worst-case ratio between the performance of the online algorithm and the offline optimal algorithm. However, the competitive ratios in many current studies are relatively low and thus cannot satisfy the need of the customers in practical applications. To provide a better service, a practice for service provider is to add more redundancy to the system. Thus we have a new problem which is to quantify the relation between the amount of increased redundancy and the system performance.

In this dissertation, to address the problem that the competitive ratio is not satisfactory, we ask the question: How much redundancy should be increased to fulfill certain performance guarantee? Based on this question, we will define a new competitive ratio showing the relation between the system redundancy and performance of online algorithm compared to offline algorithm. We will study three applications in network applications. We propose online algorithms to solve the problems and study the competitive ratio. To evaluate the performances, we further study the optimal online algorithms and some other commonly used algorithms as comparison.

We first study the application of online scheduling for delay-constrained mobile offloading. WiFi offloading, where mobile users opportunistically obtain data through WiFi rather than through cellular networks, is a promising technique to greatly improve spectrum efficiency and reduce cellular network congestion. We consider a system where the service provider deploys multiple WiFi hotspots to

offload mobile traffic with unpredictable mobile users' movements. Then we study online job allocation with hard allocation ratio requirement. We consider that jobs of various types arrive in some unpredictable pattern and the system is required to allocate a certain ratio of jobs. We then aim to find the minimum capacity needed to meet a given allocation ratio requirement. Third, we study online routing in multi-hop network with end-to-end deadline. We propose reliable online algorithms to schedule packets with unpredictable arriving information and stringent end-to-end deadline in the network.

DEDICATION

To you as a reader.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of Professor I-Hong Hou and Professors P. R. Kumar and Le Xie of the Department of Electrical and Computer Engineering, and Professor Radu Stoleru of the Department of Computer Science.

All work conducted for the dissertation was completed by the student independently.

Funding Sources

Graduate study was supported in part by the U.S. Army Research Laboratory and the U.S. Army Research Office under Grant W911NF-15-1-0279 and in part by the NPRP of Qatar National Research Fund (a member of Qatar Foundation) under Grant 8-1531-2-651.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

A major theme of this thesis is the trade-off between system resource augmentation and performance guarantee. We explore online scheduling problems in different network applications and quantify how much system redundancy is needed to achieve certain required performance.

Offline algorihtm makes decision when the entire inputs are available. In contrast, when inputs are given one by one and decision is needed upon each input, we use online algorithm to solve the problem. It is a process which makes decisions without any knowledge of future events. Competitive ratio is used to evaluated the performance of an online algorithm. It is the worst-case ratio between the performance of an online algorihtm and the offline optimal algorithm. In other words, the online algorithm guarantees such competitive ratio for any arbitrary inputs.

While the competitive ratios in existing studies reveal the performance of their proposed online algorithms, they are still not sufficient to fullfill the requirement in many practical applications. In most practical network applications, the service providers demand a highly reliable system. But none of these studies can achieve such performance. To improve the performance to meet the stringent requirement of network applications, the current practice is to increase system redundancy, since online algorithm cannot achieve better performance without knowing future inputs. Thus, besides studying the competitive ratio of online algorithms, we look further to the following question: How much redundancy increasement is needed to achieve certain required system performance? To answer this question, we define a new competitive ratio which quantifies the relationship between the system redundancy and system performance. In different applications, we propose online

algorithms and use this new competitive ratio to analyze them. We also compare our online algorithms with other commonly used algorithms. And by using this new competitive ratio, it is easy to compare how much system redundancy is needed for each different algorithm to guarantee the same system performance.

In this thesis, we study three network applications including online schuedling for delayed mobile offloading, online job allocation with hard allocation ratio requirement, and online routing in multi-hop netowrk with end-to-end deadline.

In Section 2, we study the application of delayed mobile offloading. The goal is to offloading traffic through WiFi to reduce the mobile traffic. We consider the downlink scenario where each mobile user needs to obtain some data from service provider before a certain deadline. There are multiple WiFi access points (AP) in the system, and each AP makes decision on which user to serve. Users are moving around in the system with unknown pattern, which result in change of WiFi service AP and service rate. We aim to use WiFi as much as possible, and will only use cellular networks to obtain their needed data after their respective deadlines. We design scheduling policies for WiFi APs that maximize the amount of data offloaded to WiFi. In order to reduce cellular network traffic, the service providers are willing to increase the capacity of WiFi to meet a hard requirement on the amount of data offloaded by WiFi. We then study the amount of capacity needed to provide such WiFi offload guarantees for online scheduling policies.

In section 3, we study the application of online job allocation. Such a problem exists in many emerging Internet services, such as YouTube, Netflix, etc. Each server only serve certain types of jobs and has a limited capacity. When a job arrives, it reveals the subset of servers which can serve this job and only one server will serve the job. The system makes decision on which server should the job be allocated to. A server can provide service to a new job as long as it has not reached

its capacity. If no server can process this job, the job is dropped, which can significantly impact user satisfaction. The job arrival pattern is unknown and unable to predict, thus we need online policy which makes decisions solely based on current system state. Current online algorithms cannot provide satisfactory allocation ratio. To meet the users' demand, service provider is willing to add redundancy and increase the capacity of data centers to accommodate unpredictable patterns of job arrivals. We focus on the problem of how much capacity is needed to guarantee the required job allocation ratios. We propose two simple online algorithms and derive closed-form expressions for their performance. We prove that to achieve same allocation ratio, no online policy can guarantee to use less capacity than our proposed policies.

Next, we consider the problem with job arrival and departure. In this setting, jobs arrive at arbitrary time in a slotted time system. Each job has a deadline which is revealed upon arrival. Each server has a limited serving capacity and buffer capacity. A job can be allocated to server when the server can serve this type of job and server buffer is not full. Jobs that are allocated to buffers or not served before deadlines are dropped from the system. We propose an online policy and study its performance with augmentation of both serving capacity and buffer capacity.

In section 4, we study the application of multi-hop network with end-to-end deadline. In a multi-hop network, packets arrive the network from any node at any time. Upon its arrival, it reveal the destination node and its deadline. When packets arrival patterns are unknown in advance, it is very difficult to provide performance guarantee within the constraint of end-to-end delay. A scheduling for an arriving packet is a sequence of hops and the corresponding time to use the hops. Each decision on hop-time pair has impact on the future decisions of hop-time

pairs. Current studies cannot provide a good packt deliver rate, especially when the network size is large. To maintain desirable performance using online algorithms, current practice is to add redundancy into the system. We increase the capacity of each hop and determine the amount of capacity redundancy needed to provide the desirable performance guarantees. We propose an centralized low complexity online algorithm which achieves desirable performance guarantees with a small amount of redundancy. Also we propose a heuristic distributed implementation of our centralized online algorithm. In the distributed algorithm, each node has real-time link information of its connected links. The network information on all links is broadcasted periodically. Source node suggests a scheduling decision based on the previous received information. During the packet transmitting, the route selected by the source node is fixed, but each node still has certain freedom to choose transmit time based on the real-time link information.

The main issue we address in this thesis is: how much system resource we need so that we can guarantee the system performance. We propose efficient online algorithms and compare the performance with optimal online policies and commonly used policies. We have shown that our policies can achieve the optimal bound and also outperform commonly used policies.

2. ONLINE SCHEDULING FOR DELAYED MOBILE OFFLOADING*

2.1 Problem Overview

With the increasing number of smart phone users subscribing to 3G/4G networks, the mobile data traffic grows rapidly in recent years. The global mobile data traffic growth rate in 2013 exceeds 81%, and is expected to grow at a 61% compound annual growth rate (CAGR) from 2013 to 2018 [1]. Cellular networks face the challenge of serve this great increase in data consumption.

Since interference between links is the major obstacle to dramatically increasing the capacity of wireless networks, many studies have been proposed to migrate traffic from the high-power and high-interference macro base stations to networks with smaller transmission power and interference, such as femtocells [2], WiFi [3], and mobile-to-mobile opportunistic networks [4]. Offloading traffic through WiFi has been shown to be an effective way to reduce the mobile traffic [5] [3]. WiFi is faster and uses less energy to transmit data when there is a connection [5]. Thus WiFi can significantly reduce the mobile traffic through macro base stations in the next several years. For instance, 45% of the global mobile traffic is offloaded using WiFi in 2013, and the rate is estimated to raise to 52% in 2018 [1].

In this section, we study the problem of using WiFi for delayed mobile offloading [6–8].In delayed mobile offloading, a large amount of mobile users need to obtain delay-tolerant data,such as Dropbox synchronization and App updates, from service providers. Each mobile user sets a deadline for its data, and opportunistically obtains these data through WiFi whenever it is connected to WiFi access

*Reprinted with permission from "Online scheduling for delayed mobile offloading" by Han Deng, I-Hong Hou, 2015, IEEE INFOCOM; from "On the capacity-performance trade-off of on-line policy in delayed mobile offloading" by Han Deng, I-Hong Hou, © 2017, IEEE Transactions on Wireless Communications. [9, 10]

points (APs), so as to reduce traffic on cellular networks. Due to its own mobility patterns, a mobile user may only have intermittent WiFi connections. If a user fails to obtain all its data by the deadline, it downloads the remaining data through cellular networks [9, 10].

When multiple users are connected to one WiFi AP, the AP makes decision on which user to serve. We aim to design scheduling policies for WiFi APs that maximize the amount of data offloaded to WiFi. We focus on WiFi offloading because it has been extensively deployed. However, all our results can be directly applied to other means of mobile offloading, such as offloading through femto cell networks.

We show that the problem of maximizing the amount of offloaded data can be formulated as a linear programming problem, and an offline policy can solve it with standard linear programming techniques. However, such a formulation requires the knowledge of mobility patterns of all mobile users in advance. Instead, we study the performance of online scheduling polices that make scheduling decisions only based on system history and the current locations of users. When all APs use the same transmission rates for any connected users, we show that any work-conserving scheduling policy is able to offload at least 50% as much data as the optimal offline policy. On the other hand, when APs may use different transmission rates for different users based on their individual channel qualities, we propose a simple online algorithm that guarantees to deliver at least $\frac{e-1}{e}$ as much data as the optimal offline policy when the requested data amount is large , where e is Euler's constance.

A fraction of $\frac{e-1}{e}$ of data offloaded to WiFi may not be sufficient to reduce the congestion. Hence, we further investigate the case when wireless service providers have a hard requirement on the amount of data offloaded to WiFi, so as to re-

duce cellular network congestion, and they are willing to increase the capacity of WiFi to meet this requirement. We then study the amount of capacity needed to provide offload guarantees for online scheduling policies. We propose a simple online scheduling policy and prove that, in order to offload at least $\frac{1}{\beta}$ as much data as the optimal offline policy, our policy needs to increase the capacity by approximated $\frac{1}{2(\beta-1)}$. The value of β is chosen by the service provider based on its required offloading guarantees. On the other hand, even when APs only use a fixed transmission rate, the other commonly-used round-robin, max-weight, and proportional fair policies need to increase the capacity by at least $\frac{1}{\beta-1}$ to provide the same guarantee. In other words, our policy only needs half as much capacity to provide the same performance guarantee. We further prove that no policy can guarantee offloading $\frac{1}{\beta}$ data with less than $\frac{1}{2(\beta-1)}$ capacity, and therefore our policy achieves the optimal trade-off between capacity and performance.

Theoretical analysis only shows that the worst-case performance of our policies is better than that of the three commonly-used policies. We further conduct simulations to evaluate the performance of scheduling policies for a randomly generated system. Simulation results show that our policies still outperform the other three on average. The fact that our policy is significantly better than these widely used policies in WiFi scheduling, in terms of both theoretical bounds and simulation results, further highlights that delayed WiFi offloading is fundamentally different problem from traditional WiFi scheduling.

2.2 Related Work

Many experimental studies have shown that mobile offloading is promising. Gass and Diot [11] compare WiFi and 3G network through experiments and show that WiFi is able to download more data than 3G network even if though con-

necting time is shorter. Balasubramanian et al. [12] study the availability of 3G and WiFi network from moving cars in three cities, and find that WiFi suffers greatly from limited connectivitly. They then propose a system called Wiffler to significantly improve the amount of offloaded traffic. Lee et al. [3] study the WiFi offload performance through an experiment with 100 iPhone users in Seoul, and observe that WiFi can upload about 65% of the traffic. Mota et al. [13] study the WiFi hotspots availability during bus routes in Paris, and show that current WiFi in Paris can offload up to 30% of mobile traffic. Additional work such as [14, 15] study the AP side. Dimatteo et al. [14] study how many APs are required to cover a metropolitan area offloading. Trestian et al. [15] propose to upgrade the network capacity in a selected number of locations, called Drop Zone. They design infrastructure placement algorithm which tries to reduce the AP number. It shows that by upgrading less than 1000 infrastructures across US will upload 50% of data. However, there is still no research on upgrading the APs to guarantee the offload data ratio.

Surveys in [16, 17] provide some results on how much time users are willing to wait for different applications. Lee et al. [18] analyze how much economy benefit can be generate by delayed offloading and uses real traces for numerical analysis. Mehmeti and Spyropoulos [6] use a queueing model for delayed mobile offloading and analyze the mean delay as a function of number of users and AP availability. Cai and etc. [7] propose an mechanism to encourage users to participate in delayed WiFi offloading by reward. Thus the delayed mobile offloading problem is actually worth considering.

An important challenge for mobile offloading is the unknown mobility patterns of mobile users. There are several studies that focus on deriving models for mobility patterns [19–21]. Cheung and Huang [8] study the WiFi offloading problem by

formulating the problem as a finite-horizon Markov decision process by using the prediction in [21]. Li et al. [22] study using a small set of mobile users to offload data, and propose a policy based on submodular optimization. Whitbeck et al. [23] consider using offloading to reduce the burden in broadcasting messages. Hou et al. [24] propose a transport layer protocol to integrate 3G and WiFi networks for vehicular network access. Barbieri et al. [25] propose a system design for mobile offloading with pico base stations. Bennis et al. [26] and Singh et al. [27] consider the problem of network self-organizing for offloading traffic. Bilgir Yetim and Martonosi [28] propose offline scheduling policies for WiFi offloading. These studies assume that user mobility follows some well-defined random process. In real life, user mobility may be non-ergodic, and these studies cannot be applied. In contrast, our work aims to maximize the total offloaded data without any assumptions on user mobility.

2.3 System Model

We consider a system where mobile users move within the area of a cellular network. In order to reduce the congestion of the cellular network, the cellular operator deploys a number of WiFi hotspots within the region. We use $\mathcal{I}$ to denote the set of mobile users and $\mathcal{M}$ to denote the set of WiFi APs. Mobile users may enter the system at different times and at different locations. Upon entering the system, a mobile user i specifies the amount of data, denoted by C_i, that it needs to obtain, and a deadline T_i. The mobile user moves around the system and tries to obtain data from WiFi APs whenever possible. At time T_i, the mobile user downloads all the remaining data from the cellular network directly.

We assume that time is slotted and numbered as $t = 1, 2, \ldots$. The location of a mobile user may change from time to time and it determines the connectivity

and channel capacity between APs and itself. Since APs do not have users' location information in advance, channel conditions are unpredictable. Each AP makes scheduling decisions based on the past transmission history and current channel conditions.

We use K_{imt} to denote the channel capacity between AP m and mobile user i at time t. If i cannot be connected to m at time t, we have $K_{imt} = 0$. There have been some advancements in multi-homing, where a mobile user can be connected to multiple APs simultaneously. Our model can easily accommodate multi-homing by allowing a user i to have $K_{imt} > 0$ for multiple APs. We assume each user can be connected to at most H APs at any given time.

Also, since i cannot download any data prior to its entrance, and it will use the cellular network to download data after its deadline, we set $K_{imt} = 0$ for all t prior to i's entrance or after its deadline. We normalize the system so that $0 \leq K_{imt} \leq \frac{1}{H}$ for all i, m, t. Therefore, at each time t, each client can at most obtain one unit of data.

Once client i is connected to AP m at the beginning of time slot t, the time slot is fully occupied by client i, regardless whether or not AP m is transmitting data to client i. AP m employs some scheduling policy to determine the portion of time it spends transmitting to mobile user i during time slot t, denoted by X_{imt}. The amount of data that mobile user i obtains from AP m at time t is then $K_{imt}X_{imt}$. Our goal is to design a scheduling policy that maximizes the total amount of data that are delivered through WiFi, which, in turn, minimizes the amount of data through the congested cellular network. Since each mobile user i needs to obtain C_i data, we formulate the following linear programming problem:

Offload:

$$Max \sum_{imt} X_{imt} K_{imt} \tag{2.1}$$

$$s.t. \sum_{mt} X_{imt} K_{imt} \leq C_i, \forall i \in \mathcal{I}, \tag{2.2}$$

$$\sum_{i} X_{imt} \leq 1, \forall m \in \mathcal{M}, t, \tag{2.3}$$

$$X_{imt} \geq 0, \forall i \in \mathcal{I}, m \in \mathcal{M}, t. \tag{2.4}$$

We use Γ_{opt} to denote the optimal value of $\sum_{imt} X_{imt} K_{imt}$ in the above problem. While this problem can be solved by standard linear programming techniques, doing so requires the knowledge of the entrance times and locations of all mobile users at time 0, which is impractical. Instead, we aim to derive online policies that choose the values of X_{imt} solely based on system history up to time t. We use η to denote an online policy. We let $\Gamma_\eta(1)$ be the value of $\sum_{imt} X_{imt} K_{imt}$ under policy η, given K_{imt} and C_i.

We assume that the cellular operator may be able to increase the capacity of WiFi hotspots by, for example, upgrading APs or obtaining more spectrum. When the capacity of WiFi hotspots is increased by R, the channel capacity between i and m at time t becomes RK_{imt}. Equivalently, we can also describe the system as one with channel capacity K_{imt}, but the AP can spend an amount of R time transmitting to clients in each slot, that is, $\sum_i X_{imt} \leq R$. Therefore, we consider the following linear programming problem when the capacity is increased by R:

Offload(R):

$$Max \sum_{imt} X_{imt} K_{imt} \tag{2.5}$$

$$s.t. \sum_{mt} X_{imt} K_{imt} \leq C_i, \forall i \in \mathcal{I}, \tag{2.6}$$

$$\sum_{i} X_{imt} \leq R, \forall m \in \mathcal{M}, t, \tag{2.7}$$

$$X_{imt} \geq 0, \forall i \in \mathcal{I}, m \in \mathcal{M}, t. \tag{2.8}$$

Let $\Gamma_\eta(R)$ be the value of $\sum_{imt} X_{imt} K_{imt}$ for the Offload(R) problem under policy η. We evaluate the performance of η by its *competitive ratio*, which is defined slightly differently from most existing literature.

Definition 1. *A policy η is said to be (R, θ)-competitive if $\Gamma_{opt}/\Gamma_\eta(R) \leq \theta$, as $\min_{i \in \mathcal{I}} C_i \to \infty$, for all systems.*

We note that when $R = 1$, the corresponding β becomes the competitive ratio commonly defined in existing literature. Our definition is richer in that it characterizes the amount of capacity needed to provide performance guarantees. Since the very reason of using WiFi offloading is that the cellular network is congested, the operator may have a hard requirement on the amount of data being offloaded through WiFi, and it is willing to purchase better equipments and more spectrum to achieve this requirement. In this case, it needs to know how much capacity is needed. Suppose the optimal offline policy can offload all data through WiFi, and the operator requires a portion $1/\theta$ of the data to be offloaded, our definition then reveals that the capacity needs to be increased by R so that the employed policy is (R, θ)-competitive.

We summarize the notations we used in the section in Table 2.1.

Table 2.1: Notations
Reprint with permission from [10].

$\mathcal{I}$	Mobile user set
i	Single user
$\mathcal{M}$	WiFi AP set
m	Single AP
t	Time slot
K_{imt}	Normalized channel capacity between AP m and user i at time t
X_{imt}	Portion of time AP m spends on user i at time slot t
H	The maximum number of APs that a user can connect to
C_i	Data amount that user i needs
T_i	Time that user i switch from WiFi to cellular network
Y_{mt}, Z_i	Dual variables
η	Policy
R	Capacity increasing rate
Γ_{opt}	Optimal value of (2.1) in **Offload**
$\Gamma_\eta(1)$	Value of (2.1) in **Offload** under policy η
$\Gamma_\eta(R)$	Value of (2.5) in **Offload(R)** under policy η
θ	Value of $\max(\Gamma_{opt}/\Gamma_\eta(R))$

2.4 Preliminary

This section introduces some basic theorems that will be used.

A standard form of linear programming problem (LP) is:

$$
\begin{aligned}
(P) : Max & \sum_{i=1}^{n} c_i x_i, \\
s.t. & \sum_{i=1}^{n} a_{ij} x_i \leq b_j, \forall 1 \leq i \leq n, \\
& x_i \geq 0,
\end{aligned}
$$

and its dual is

$$(D) : Min \sum_{j=1}^{m} b_j y_j,$$
$$s.t. \sum_{j=1}^{m} a_{ij} y_j \geq c_i, \forall 1 \leq j \leq m,$$
$$y_j \geq 0.$$

We have the following two fundamental theorems:

Theorem 1 (Weak Duality [29]). *Let $\{x_i\} \in \mathbb{R}^n$ and $\{y_j\} \in \mathbb{R}^m$ satisfy the constraints of the primal (P) and the dual (D) LPs, respectively, then:*

$$\sum_{i=1}^{n} c_i x_i \leq \sum_{j=1}^{m} b_j y_j.$$

Theorem 2 (Complementary Slackness [29]). *Let $\{x_i\} \in \mathbb{R}^n$ and $\{y_i\} \in \mathbb{R}^m$ satisfy the constraints of the primal (P) and dual (D) LPs, respectively. Further, $\{x_i\}$ and $\{y_i\}$ have the following properties:*

- *If $x > 0$, then $c_i \leq \sum_{j=1}^{m} a_{ij} y_i \leq \beta \cdot c_i$ for some $\beta > 1$;*
- *If $y > 0$, then $\sum_{i=1}^{n} a_{ij} x_i = b_j$;*

Then:

$$\sum_{j=1}^{m} b_j y_j \leq \beta \cdot \sum_{i=1}^{n} c_i x_i.$$

2.5 Competitive Ratio with Unit Capacity

In this section, we discuss the special case with $R = 1$. We first show that when K_{imt} is either 0 or 1, any work-conserving policy is $(1, 2)$-competitive. We then study the case when K_{imt} can be any real number in $[0, 1]$. We propose a simple online scheduling policy and prove that it is $(1, \frac{e}{e-1})$-competitive.

2.5.1 Performance of Work-Conserving Policy under On-Off Channels

We first consider the case where K_{imt} is either 0 or 1, which is usually referred as *On-Off channels*, and we say that client i is *connected* to AP m at time t if $K_{imt} = 1$. We study the performance of *work-conserving scheduling policy*, under which each AP m selects to serve one connected client that has yet to receive all the data, as long as there is one, and only idles when all connected clients have already received all their data.

Theorem 3. *Any work-conserving policy is $(1, 2)$-competitive with ON-Off channels.*

Proof. The offload problem is shown as (2.1) to (2.4), and its dual is

$$(D) : Min \sum_{mt} Y_{mt} + \sum_{i} C_i Z_i, \tag{2.9}$$

$$s.t.\ Y_{mt} + K_{imt} Z_i \geq K_{imt}, \forall i, m, t, \tag{2.10}$$

$$Y_{mt} \geq 0, \forall m, t, \tag{2.11}$$

$$Z_i \geq 0, \forall i, \tag{2.12}$$

where Y_{mt} is the dual variable for each constraint in (2.2), and Z_i is the dual variable for each constraint in (2.3).

We set $X_{imt} = 1$ if client i is served by AP m at time slot t, and $X_{imt} = 0$

otherwise. We set $Y_{mt} = 1$ if AP m schedules a client at time t, and $Y_{mt} = 0$ if m idles at t. We set $Z_i = 1$ if client i have received all its data before its deadline, and $Z_i = 0$ otherwise.

We will use Theorem 2 to establish the theorem. First, we show that X_{imt}, Y_{mt}, and Z_i satisfy the constraints (2.2) (2.3), and (4.10). (2.2) and (2.3) are satisfied because each AP schedules at most one client at any time, and it never schedules clients that have already received all their data.

Given i, m, t, if $K_{imt} = 0$, then (4.10) is satisfied since Y_{mt} and Z_i are non-negative. (4.10) also holds if $K_{imt} = 1$ and $Y_{mt} = 1$. Finally, if $K_{imt} = 1$ and $Y_{mt} = 0$, i.e., AP m does not schedule any client at time t, then all clients connected to AP m at time t must have already received all their data. Hence, $Z_i = 1$ and (4.10) still holds.

Next, we verify the complementary slackness conditions. If $Z_i > 0$, then client i obtains all its data, and $\sum_{mt} X_{imt}K_{imt} = C_i$. If $Y_{mt} > 0$, then AP m schedules some client at time t, and $\sum_i X_{imt} = 1$. In addition, if $X_{imt} > 0$, then $K_{imt} = 1$. Thus, $2K_{imt} = 2 \geq Y_{mt} + K_{imt}Z_i \geq K_{imt}$. By Theorem 2, we know $\sum_{mt} Y_{mt} + \sum_i C_i Z_i \leq 2 \cdot \sum_{imt} X_{imt}K_{imt}$. Further, $\Gamma_{opt} \leq \sum_{mt} Y_{mt} + \sum_i C_i Z_i$, by Theorem 1, and hence any work-conserving policy is $(1, 2)$-competitive. □

2.5.2 Online Algorithm for General Channels

We now discuss the general case in which K_{imt} can be any real number between 0 and 1. We propose an online scheduling algorithm and prove that it is $(1, \frac{e}{e-1})$-competitive.

In our algorithm, APs keep track of and update a variable Z_i for each client i. Z_i is initially set to 0. If each time t, each AP m chooses to serve the client i that maximizes $K_{imt}(1 - Z_i)$, and delivers K_{imt} data to the client. Each time

client i obtains K_{imt} data from an AP m, Z_i will be updated as $Z_i(1 + \frac{K_{imt}}{C_i}) + \frac{K_{imt}}{(d-1)C_i}$. At time t, if there are multiple APs that serve i at the same time t, which is possible under multi-homing, Z_i will be updated as $Z_i(1 + \frac{\sum_{m:m \text{ serves } i} K_{imt}}{C_i}) + \frac{\sum_{m:m \text{ serves } i} K_{imt}}{(d-1)C_i}$. Here d is a value only used in calculation and it is set to be $(1 + 1/C_{min})^{C_{min}}$. We show the value chosen for d is reasonable in proof of Lemma 1. AP m broadcasts the updated Z_i to all APs. Algorithm 1 formally describes the algorithm. In Algorithm 1, we also introduce two other variables, X_{imt} and Y_{mt}. These two variables are only used to establish the competitive ratio, and are not needed in actual implementations.

In Algorithm 1, each of Y_{mt} and X_{imt} is only updated at time slot t, while Z_i may be updated in many different time slots. We note that the value of Z_i is non-decreasing in each update.

At time t, it is possible that i already obtains most of its data and only needs less than $\sum_{m:m \text{ serves user } i} K_{imt}$ data to complete its download. In this case, APs use only a fraction of a time slot to deliver all remaining data that i needs. Step 14 addresses this case, and the total amount of offloaded data is $\sum_{imt} X_{imt} K_{imt}$.

Lemma 1. *Let $Z_i^{(t)}$ be the value of Z_i at the end of time slot t. Then,*

$$Z_i^{(t)} \geq (\frac{1}{d-1})(d^{\sum_{m,s \leq t} \frac{X_{imt} K_{imt}}{C_i}} - 1). \tag{2.13}$$

Proof. We prove (2.13) by induction on t.

When $t = 0$, $Z_i^{(t)} = 0 = (\frac{1}{d-1})(d^0 - 1)$, and (2.13) holds.

Suppose (2.13) holds for all time before s. Consider time $t = s + 1$. If i is not

Algorithm 1 Policy for unit capacity

Initially, $X_{imt} = 0$, $Y_{mt} = 0$, $Z_i = 0$.
$C_{min} \leftarrow \min_i C_i$, $d \leftarrow (1 + 1/C_{min})^{C_{min}}$.
for each time slot t **do**
  for each AP m **do**
    $i_m^* \leftarrow \underset{i}{\text{argmax}}\{\sum_{m:m \text{ serves } i} K_{imt}(1 - Z_i)\}$.
    if $K_{i_m^* mt}(1 - Z_{i_m^*}) > 0$ **then**
      $Y_{mt} \leftarrow K_{i_m^* mt}(1 - Z_{i_m^*})$.
      $X_{i_m^* mt} \leftarrow 1$.
    end if
  end for
  for each client i **do**
    $Z_i \leftarrow Z_i(1 + \frac{\sum_{m:m \text{ serves } i} K_{ipt}}{C_i}) + 1\frac{\sum_{m:m \text{ serves } i} K_{ipt}}{(d-1)C_i}$.
    if $\sum_{p,s \le t} X_{ips} K_{ips} > C_i$ **then**
      $X_{imt} \leftarrow \frac{C_i - \sum_{p,s<t} X_{ips} K_{ips}}{\sum_{p:p \text{ serves } i} K_{ipt}}$, $\forall$ m servers i
    end if
  end for
end for

scheduled at $s+1$, $X_{imt} = 0$ for all m at $t = s+1$ and $Z_i^{(s+1)} = Z_i^{(s)}$. Hence (2.13) holds.

On the other hand, if i is scheduled by AP p at time $s+1$,

$$
\begin{aligned}
& Z_i{}^{(s+1)} \\
=& Z_i^{(s)}(1 + \frac{\sum_{p:p \text{ serves } i} K_{ip(s+1)}}{C_i}) + \frac{\sum_{p:p \text{ serves } i} K_{ip(s+1)}}{(d-1)C_i} \\
\ge& \frac{1}{(d-1)}(d^{\sum_{m,t \le s} \frac{X_{imt} K_{imt}}{C_i}} - 1)(1 + \frac{\sum_{p:p \text{ serves } i} K_{ip(s+1)}}{C_i}) \\
& + \frac{\sum_{p:p \text{ serves } i} K_{ip(s+1)}}{(d-1)C_i} \\
=& \frac{1}{(d-1)}[d^{\sum_{m,t \le s} \frac{X_{imt} K_{imt}}{C_i}}(1 + \frac{\sum_{p:p \text{ serves } i} K_{ip(s+1)}}{C_i}) - 1]
\end{aligned}
$$

It is easy to verify that $ln(1+x)/x$ is decreasing when $x \in [0,1]$. Thus $(1+y) \geq (1+x)^{(y/x)}$ for $x \geq y$. Let $y = \dfrac{\sum_{p:p \text{ serves } i} K_{ip(s+1)}}{C_i}$ and $x = \dfrac{1}{C_{min}}$. We then have

$$
\begin{aligned}
& {Z_i}^{((s+1)} \\
\geq & \frac{[(d^{\sum_{m,t \leq s} \frac{X_{imt} K_{imt}}{C_i}})(1+\frac{1}{C_{min}})^{\frac{\sum_{p:p \text{ serves } i} K_{ip(s+1) C_{min}}}{C_i}} - 1]}{(d-1)}
\end{aligned}
$$

Recall that the value of d is $(1+1/C_{min})^{C_{min}}$. Thus

$$
{Z_i}^{(s+1)} \geq \frac{1}{(d-1)}(d^{\sum_{m,t \leq s+1} \frac{X_{imt} K_{imt}}{C_i}} - 1),
$$

and (2.13) holds. By induction, (2.13) holds for all t. □

Theorem 4. *Algorithm 1 is* $(1, \frac{e}{e-1})$*-competitive.*

Proof. The offload problem and and its dual are stated as (2.1) to (2.4), and (4.9) to (4.12), respectively. We prove Algorithm 1 is $(1, \frac{e}{e-1})$-competitive by the following steps:

First, we show that the dual solutions $\{Y_{mt}\}$ and $\{Z_i\}$ satisfy constraints (4.10) to (4.12).

Since $i_m^* \leftarrow \underset{i}{\text{argmax}}\{K_{imt}(1-Z_i)\}$, we have:

$$
K_{i_m^* mt}(1 - Z_{i_m^*}) \geq K_{imt}(1 - Z_i), \forall i, m, t.
$$

Further, by step 7 in Algorithm 1, we have:

$$\begin{aligned}
&Y_{mt} + K_{imt}Z_i - K_{imt} \\
\geq &K_{i^*_m mt}(1 - Z_{i^*_m}) + K_{imt}Z_i - K_{imt} \\
\geq &K_{imt}(1 - Z_i) + K_{imt}Z_i - K_{imt} = 0.
\end{aligned}$$

Thus (4.10) is satisfied. It is easy to check that Y_{mt} and Z_i are non-negative, and (4.11) and (4.12) hold.

Second, we show that X_{imt} satisfy constraints (2.2) to (2.4). Step 14 ensures that (2.2) holds. By Lemma 1, $Z_i^{(t)} < 1$ only when i does not receive all its data at time t. Hence, X_{imt} is updated only if the total received data of client i is less than its C_i, which is $\sum_{p,s<t} X_{ips}K_{ips} < C_i$, which makes (2.3) and (2.4) hold.

Third, we show that every time steps 7, 8, and 12 are invoked, the ratio between the change of the dual objective function (4.9) and change of the primal objective function (2.1) is $\frac{d}{d-1}$. We note that we ignore the change of (2.1) by step 14 now, which will be taken into account later.

When AP m schedules i at time t, X_{imt} is increased from 0 to 1, and (2.1) is increased by $\sum_{m:m \text{ serves } i} K_{imt}$. Meanwhile, (4.9) is increased by

$$\begin{aligned}
&\sum_{m:m \text{ serves } i} K_{imt}(1 - Z_i^{(t-1)}) + C_i(Z_i^{(t-1)} \frac{\sum_{m:m \text{ serves } i} K_{imt}}{C_i} \\
&+ \frac{\sum_{m:m \text{ serves } i} K_{imt}}{(d-1)C_i}) \\
=&(1 + \frac{1}{d-1}) \sum_{m:m \text{ serves } i} K_{im_j t}.
\end{aligned}$$

Thus the ratio between change of (4.9) and (2.1) is $1 + \frac{1}{d-1} = \frac{d}{d-1}$.

Let Γ_{opt} be the optimal value of (2.1), $\Gamma_{dual,\eta}$ be the value of (4.9) under Algorithm 1, and $\Gamma^*_{prim,\eta}$ be the value of (2.1) under Algorithm 1 without step 14. We have established that $\Gamma_{opt} \leq \Gamma_{dual,\eta} = \frac{d}{d-1}\Gamma^*_{prim,\eta}$, where $\Gamma_{opt} \leq \Gamma_{dual,\eta}$ because of Theorem 1.

Finally, we address the influence of step 14. Step 14 is only invoked when i^*_m obtains all its data, i.e., $\sum_{p,s} X_{i^*_m ps} K_{i^*_m ps} = C_{i^*_m}$. By Lemma 1, step 14 is invoked at most for one time slot for each client. Further, when step 14 is invoked, (2.1) decreases by no more than $\sum_{m:m \text{ serves } i} K_{imt} \leq 1$. Let $\Gamma_{prim,\eta}$ be the value of (2.1) under Algorithm 1 with step 14. We now have $\Gamma_{prim,\eta} \geq \Gamma^*_{prim,\eta}(1-\frac{1}{C_{min}}) \geq \Gamma_{opt}\frac{d-1}{d}(1-\frac{1}{C_{min}})$. Since $d \to e$, as $C_{min} \to \infty$, Algorithm 1 is $(1, \frac{e}{e-1})$-competitive. □

2.6 Competitive Ratio for Variable Capacity

In the previous section, we obtain a $(1, \frac{e}{e-1})$-competitive online algorithm. Thus, Algorithm 1 guarantees to offload 63% as much data as an optimal offline algorithm does. However, this also indicates that, when the optimal offline algorithm offloads all data, our algorithm may miss almost 37% of the data. Then, how much capacity is needed to guarantee offloading, say, 95% of the data? We will focus on this problem in this section.

It is first of interests to study whether it is feasible to increase the capacity by R times so as to guarantee an onine algorithm can always offload as much data as an optimal offline algorithm with unit capacity does. Or, with our terminology, to study whether there exists a $(R, 1)$-competitive policy. The following example shows that $(R, 1)$-competitive policy does not exist, for any R.

Example 1. *Fix R. Consider a system consisting of $N = R + 1$ clients and one AP with On-Off channels. Each client has a file size of C. One of the clients, say, client 1, enters the AP coverage area at time 1 and leaves at time C, while all other clients*

enter the AP coverage area at time 1 and stay forever. We have $T_i = \infty$ for all clients. The optimal offline policy schedules client 1 in times $1 \leq t \leq C$, and then schedules other clients after $t = C$. Hence, the optimal offline policy offloads all data. On the other hand, since online policies do not know which client is connected to the AP only in times $1 \leq t \leq C$, and they can at most offload $RC < NC$ data in times $1 \leq t \leq C$, they cannot guarantee to offload all data. □

When the capacity is increased by R, the corresponding offload problem is described in (2.5)–(2.8). In this section, we first propose two online policies and study their competitive ratios. We then derive a theoretical lower-bound for the competitive ratio of all online policies. Finally, we study the competitive ratio of the round robin policy.

2.6.1 Primal-Dual Scheduling Policy

The primal-dual (PD) scheduling policy is very similar to Algorithm 1. It is described as Algorithm 2. The only differences are that we choose $d = (1 + 1/C_{min})^{C_{min}/R}$, and we assign $X_{imt} = R$ if i is scheduled by m at time t.

We now study the competitive ratio of PD.

Lemma 2. *Let $Z_i^{(t)}$ be the value of Z_i at time slot t. We have*

$$Z_i^{(t)} \geq (\frac{1}{d-1})(d^{\sum_{m,s\leq t} \frac{X_{imt}K_{imt}}{C_i}} - 1). \tag{2.14}$$

Proof. We prove (2.14) by induction on t. The proof is similar to that of Lemma 1. □

Theorem 5. *PD is $(R, \frac{e^{1/R}}{R[e^{1/R}-1]})$-competitive. It is approximately $(R, 1 + \frac{1}{2R})$-competitive.*

Algorithm 2 Primal-Dual Algorithm

1: Initially, $X_{imt} = 0$, $Y_{mt} = 0$, $Z_i = 0$.
2: $C_{min} \leftarrow \min_i C_i$, $d \leftarrow (1 + 1/C_{min})^{C_{min}/R}$.
3: **for** each time slot t **do**
4: **for** each AP m **do**
5: $i_m^* \leftarrow \underset{i}{\operatorname{argmax}}\{K_{imt}(1 - Z_i)\}$.
6: **if** $K_{i_m^* mt}(1 - Z_{i_m^*}) > 0$ **then**
7: $Y_{mt} \leftarrow K_{i_m^* mt}(1 - Z_{i_m^*})$.
8: $X_{i_m^* mt} \leftarrow R$.
9: **end if**
10: **end for**
11: **for** each client i **do**
12: $Z_i \leftarrow Z_i(1 + \frac{\sum_{m:m \text{ serves } i} K_{imt}}{C_i}) + \frac{\sum_{m:m \text{ serves } i} K_{imt}}{(d-1)C_i}$.
13: **if** $\sum_{p,s \le t} X_{ips} K_{ips} > C_i$ **then**
14: $X_{imt} \leftarrow \frac{C_i - \sum_{p,s<t} X_{ips} K_{ips}}{\sum_{p:p \text{ serves } i} K_{imt}}$. $\forall m : m$ serves i
15: **end if**
16: **end for**
17: **end for**

Proof. We prove PD is $(R, \frac{e^{1/R}}{R[e^{1/R} - 1]})$-competitive by the following steps:

First, $\{Y_{mt}\}$ and $\{Z_i\}$ satisfy constraints (4.10)–(4.12). The proof is the same as the proof for Theorem 4.

Second, we show that X_{imt} satisfy constraints (2.6)–(2.8). Step 14 ensures (2.6). Further, by Lemma 2, $Z_i^{(t)} < 1$ only if $\sum_{m,s \le t} X_{imt} K_{imt} < C_i$. Therefore, a client is only scheduled when it is yet to receive all its data, which ensures (2.7) and (2.8).

Third, we show that whenever steps 7, 8, and 12 are invoked, the ratio between the change of (2.5) and the change of (4.9) is $\frac{d}{(d-1)R}$. We ignore the change of (2.5) due to step 14 now.

Suppose client i is scheduled by AP m at time t. We have $\sum_{m:m \text{ serves } i} X_{imt} = R$,

and (2.5) is increased by $R\sum_{m:m \text{ serves } i} K_{imt}$. On the other hand, (4.9) is increased by

$$\begin{aligned}
&\sum_{m:m \text{ serves } i} K_{imt}(1 - Z_i^{(t-1)}) + C_i(Z_i^{(t-1)} \frac{\sum_{m:m \text{ serves } i} K_{imt}}{C_i} \\
&+ \frac{\sum_{m:m \text{ serves } i} K_{imt}}{(d-1)C_i}) \\
=&(1 + \frac{1}{d-1}) \sum_{m:m \text{ serves } i} K_{imt}.
\end{aligned}$$

Thus the ratio between the change of objective functions (2.5) and (4.9) is $(1 + \frac{1}{d-1})/R = \frac{d}{R(d-1)}$.

Finally, we consider the influence of step 14. Step 14 is only invoked when i_m^* obtains all its data, i.e., $\sum_{p,s} X_{i_m^* ps} K_{i_m^* ps} = C_{i_m^*}$. By Lemma 2, step 14 is invoked at most once for each client. Further, when step 14 is invoked, (2.5) decreases by no more than $R\sum_{m:m \text{ serves } i_m^*} K_{i_m^* mt} \leq R$, since we normalized our system such that $K_{imt} \leq \frac{1}{H}$. Therefore, throughout the system lifetime, the ratio of decrease caused by step 14 is no more than $\frac{R}{C_{min}}$.

As $C_{min} \to \infty$, $d \to e^{1/R}$. By Theorem 1 and the above arguments, we establish that PD is $(R, \frac{e^{1/R}}{R[e^{1/R} - 1]})$-competitive.

We can approximate $\frac{e^{1/R}}{R[e^{1/R} - 1]}$ by Taylor series as follows:

$$\begin{aligned}
\frac{e^{\frac{1}{R}}}{R[e^{\frac{1}{R}} - 1]} =& \frac{1 + \frac{1}{R} + \frac{1}{2!R^2} + ...}{R(\frac{1}{R} + \frac{1}{2!R^2} + ...)} \\
=& \frac{1 + \frac{1}{R} + \frac{1}{2!R^2} + ...}{1 + \frac{1}{2!R} + ...} \\
\approx& \frac{1 + \frac{1}{R}}{1 + \frac{1}{2R}} = 1 + \frac{\frac{1}{2R}}{1 + \frac{1}{2R}} \approx 1 + \frac{1}{2R},
\end{aligned}$$

when $R \gg 1$.

Thus, the competitive ratio is approximately $(R, 1+\frac{1}{2R})$. Fig. 2.1 plots $\frac{e^{1/R}}{R[e^{1/R}-1]}$ and $1+\frac{1}{2R}$. As can be seen in the figure, $1+\frac{1}{2R}$ is a very accurate approximation even for small R.

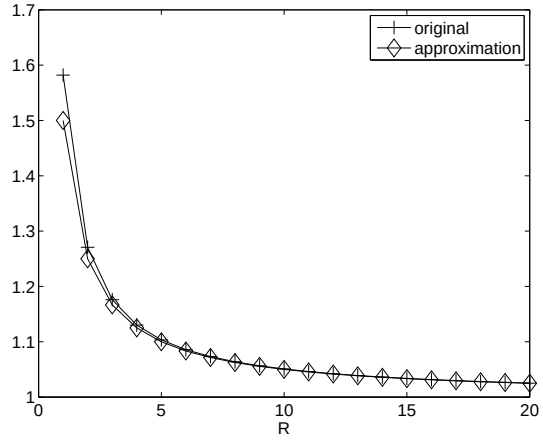

Figure 2.1: Illustration of the approximation.
Reprint with permission from [10].

□

2.6.2 Least Progress First Scheduling Policy

When implementing PD, APs need to keep track of an artificial variable Z_i and update the value of Z_i with each other. Further, PD needs to know the value of C_{min} to set d. Below, we describe an approximation of PD that is simpler and does not require any information exchange.

Using an argument similar to the proof of Lemma 2, we approximate $Z_i^{(t)}$ by $(\frac{1}{d-1})(d^{\sum_{m,s\leq t}\frac{X_{imt}K_{imt}}{C_i}}-1)$. We further approximate d by $e^{1/R}$, and e^x by $1+x$, for

all $0 < x < 1$.

With these approximations, we have

$$\begin{aligned} K_{imt}(1 - Z_i) \approx & K_{imt}[1 - (\frac{1}{d-1})(d^{\sum_{m,s<t} \frac{X_{ims}K_{ims}}{C_i}} - 1)] \\ \approx & \frac{K_{imt}}{d-1}(e^{1/R} - e^{\sum_{m,s<t} \frac{X_{ims}K_{ims}}{RC_i}}) \\ \approx & \frac{1}{(d-1)R} K_{imt}(1 - \frac{\sum_{m,s<t} X_{ims}K_{ims}}{C_i}). \end{aligned}$$

Since PD makes AP m schedule the client with the largest $K_{imt}(1 - Z_i)$, we can approximate PD by making each AP m schedule the client with the largest $K_{imt}(1 - \frac{\sum_{m,s<t} X_{ims}K_{ims}}{C_i})$. Further, we note that $(1 - \frac{\sum_{m,s<t} X_{ims}K_{ims}}{C_i})$ is the portion of data that i is yet to obtain. Therefore, this policy simply schedules the client with the largest product of channel capacity and portion of undelivered data, both values are readily available at APs. The policy is summarized in Algorithm 3 and is called *Least Progress First* (LPF). It can be easily implemented in a fully distributed fashion. Finally, we note that when all clients need to obtain the same amount of data, i.e. $C_i \equiv C$, then LPF becomes the same as the well-known Max-Weight scheduling policy. However, as we will show in Section 2.7.2, Max-Weight policy can be much worse than LPF when different clients have different C_i.

Algorithm 3 Least Progress First

1: **for** each time slot t and each AP m **do**
2: $\quad i_m^* \leftarrow \underset{i}{\text{argmax}}\{K_{imt}\frac{\text{Amount of undelivered data}}{C_i}\}$.
3: $\quad$ AP m transmits to client i_m^* at time t.
4: **end for**

2.6.3 A Lower Bound on Competitive Ratio

In the previous section, we show that the competitive ratio of our online scheduling policy is approximately $(R, 1+\frac{1}{2R})$. In this section, we are interested in the best competitive ratio that online policies can achieve.

Theorem 6. *For any given $\epsilon > 0$, there exists a positive number R_0, such that, when $R > R_0$, no policy has better competitive ratio than $(R, 1+\frac{1}{2R}-\frac{\epsilon}{R})$.*

Proof. Consider a system consisting of N clients with same file size C and one AP with On-Off channels. Assume there are k clients that will move around the area and $N-k$ clients stay in the same place. For client i, if $i \leq k$, $K_{i1t} = 1$ in time $1 \leq t \leq i \times C$, in other words, client 1 is connected to AP in time $[1, C]$; client 2 is connected to AP in time $[1, 2C]$; client 3 is connected to AP in time $[1, 3C]$; etc. If $i > k$, client i is connected to the AP forever. $T_i = \infty$ for all clients.

If the connection of all clients are known in advance, the optimal offline policy with unit capacity is to schedule the clients in the order of $1, 2, 3, ..., N$. The optimal offline policy is then able to transmit NC amount of data.

On the other hand, online policies cannot know K_{imt} in advance. Also, all clients have the same file size. Therefore, when the system capacity is increased by R times, the best that the AP can do is to evenly distribute its capacity R among all connected clients. The AP delivers a total RC amount of data in the first C time slots, and client 1 receives $\frac{RC}{N}$ amount of data. In the first $2C$ time slots, client 2 recieves $\frac{RC}{N} + \frac{RC}{N-1}$ amount of data.... In the first $k \times C$ time slots, client k recieves $\frac{RC}{N} + \frac{RC}{N-1} + ... + \frac{RC}{N-k+1}$ amount of data. The other clients receive all their data. Thus, the AP can at best deliver $\Gamma_\eta(R) = \{\frac{RC}{N}\} + \{\frac{RC}{N} + \frac{RC}{N-1}\} + ... + \{\frac{RC}{N} + \frac{RC}{N-1} + ... + \frac{RC}{N-k+1}\} + (N-k)C$ amount of data.

Let $N = (k+1)R$. Since $\frac{R}{N} < \frac{R}{N-1} < ... < \frac{R}{N-k+1}$, the competitive ratio is:

$$\begin{aligned}
\beta &= \frac{NC}{\Gamma_\eta(R)} \\
&> \frac{N}{\frac{R}{N-k+1} \cdot \frac{(1+k)k}{2} + (N-k)} \\
&= \frac{(k+1)R}{\frac{R}{(k+1)R-k+1} \cdot \frac{(1+k)k}{2} + ((k+1)R-k)} \\
&> \frac{(k+1)R}{\frac{R}{(k+1)R-(k+1)} \cdot \frac{(1+k)k}{2} + ((k+1)R-k)} \\
&= 1 + \frac{kR-2k}{2(k+1)R(R-1) - 2k(R-1) + kR} \\
&> 1 + \frac{kR-2k}{2(k+1)R^2 - 2kR + kR} \\
&> 1 + \frac{kR-2k}{2(k+1)R^2} = 1 + \frac{k-2k/R}{2(k+1)R} \\
&= 1 + \frac{1}{2R} - \frac{1}{R}(\frac{1}{2(k+1)} + \frac{k}{(k+1)R})
\end{aligned}$$

For sufficiently large k and R, we have $\frac{1}{2(k+1)} < \epsilon/2$, $\frac{k}{(k+1)R} < \epsilon/2$, and therefore $\beta > 1 + \frac{1}{2R} - \frac{\epsilon}{R}$. This completes the proof. □

Since PD is approximately $(R, 1 + \frac{1}{2R})$-competitive, Theorem 6 demonstrates that PD indeed achieves the optimal trade-off between capacity and the amount of offloaded data.

2.7 Comptitive Ratios of Other Policies

In Section 2.5.1, we have shown that the competitive ratio of any work-conserving policy is at least $(1, 2)$ with On-Off channels. In comparison, the competitive ratio of Algorithm 1 is $(1, \frac{e}{e-1}) \approx (1, 1.58)$. It appears that the competitive ratio of any work-conserving policy is close to that of Algorithm 1. We now study the competitive ratio of work-conserving policies when the capacity is increased by R times. In

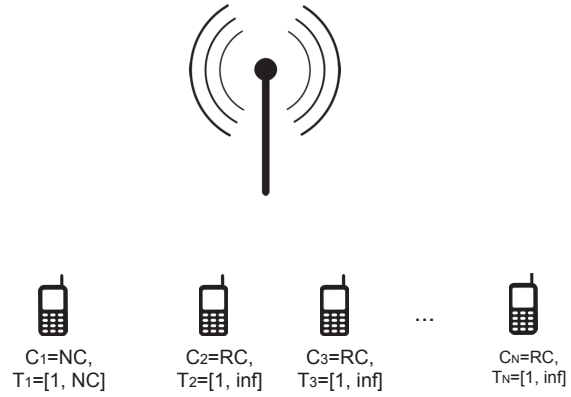

Figure 2.2: System Construction for Round Robin.
Reprint with permission from [10].

particular, we establish a lower-bound on competitive ratio for the following three policies.

2.7.1 Round Robin Scheduling Policy

With round robin policy (RR), each AP evenly distributes its capacity among all connected clients.

Theorem 7. *Round robin scheduling policy cannot have better competitive ratio than* $(R, 1 + \frac{1}{R})$.

Proof. Given R, we construct a system with one AP and N clients as follows: $C_1 = NC$, and $C_i = RC$, for all $i \neq 1$. $K_{11t} = 1$ for $1 \leq t \leq C_1$, and $K_{11t} = 0$ for $t > C_1$. For $i \neq 1$, $K_{i1t} = 1$ for all t. In other words, client 1 is connected to AP in time $[1, C_1]$, while all other clients are connected to the AP forever. $T_i = \infty$ for all clients. The system is shown is Fig. 2.2.

The optimal offline policy is first to serve client 1 from time 1 to NC. Then client 2 to N will be served. Thus the optimal offline policy is able to deliver

all $C_1 + \sum_{i=2}^{N} C_i$ amount of data with unit capacity, while round robin transmits $\frac{RC_1}{N} + \sum_{i=2}^{N} C_i$ amount of data with R capacity. The competitive ratio of round robin is then at least:

$$\begin{aligned}\beta &= \frac{C_1 + \sum_{i=2}^{N} C_i}{\frac{RC_1}{N} + \sum_{i=2}^{N} C_i} = \frac{N + R(N-1)}{R + R(N-1)} \\ &= \frac{\frac{N}{R} + (N-1)}{1 + (N-1)} \\ &\to 1 + \frac{1}{R},\end{aligned}$$

as $N \to \infty$. □

2.7.2 Max-Weight Policy

With max-weight (MW) policy, of all connected clients, each AP selects the one with maximum value of the product of K_{imt} and the client's remaining data.

Theorem 8. *The max-weight policy cannot have better competitive ratio than* $(R, 1 + \frac{1}{R})$.

Proof. Given R, we construct a system with one AP and $N + 1$ clients as follows: $C_1 = MC + C$, and $C_i = C$, for all $i \neq 1$. $K_{11t} = 1$ for all t, $K_{i1t} = 1$ for $1 \leq t \leq NC$. In other words, client 1 is connected to AP forever, while all other clients are connected to AP in time $[1, NC]$. Here we choose $N = M/R$, and $T_i = \infty$ for all clients.

The optimal offline policy is first to offload data requested by client 2 to $N + 1$ in time $[1, NC]$. Then the policy will offload data requested by client1. In this case, the policy is able to deliver all $C_1 + \sum_{i=2}^{N} C_i = MC + C + NC$ amount of data with unit capacity.

With R capacity, MW policy first transmits MC data to client 1 until it has

remaining data request of C, which is the same as the other clients. It takes NC amount of time to transmit MC amount of data. Since client i, $i \neq 1$, has a deadline of NC, they will no longer get data through WiFi. Then WiFi will transmit the remaining C data to client 1. Thus the total amount of data transmitted is $MC+C$. The competitive ratio of MW policy cannot be better than:

$$\begin{aligned}\beta &= \frac{MC + C + NC}{MC + C} = 1 + \frac{N}{M+1} \\ &\to 1 + \frac{1}{R}\end{aligned}$$

as $N \to \infty$.

□

2.7.3 Proportional Fair Scheduling Policy

With proportional fair (PF) policy, of all connected clients, each AP selects the one with the maximum value of $\frac{K_{imt}}{Throughput\ of\ client\ i}$.

Theorem 9. *The proportional fair policy cannot have better competitive ratio than* $(R, 1 + \frac{1}{R})$.

Proof. Given R, we construct a system with one AP and $N + 1$ clients as follows: $C_1 = MC$, and $C_i = C$, for all $i \neq 1$. $K_{11t} = 1$ for $1 \leq t \leq MC$, $K_{i1t} = 1$ for all t. In other words, client 1 is connected to AP in time $[1, MC]$, while all other clients are connected to AP forever. Here we choose $N = MR$, and $T_i = \infty$ for all clients..

The optimal offline policy is first to serve client 1 in time 1 to MC. After time NC, the other clients will be served. Thus the optimal offline policy is able to deliver all $C_1 + \sum_{i=2}^{N} C_i = MC + NC$ amount of data with unit capacity.

With R capacity, from time $[1, MC]$, PF policy delivers $\frac{MCR}{N+1}$ data for each client. After time MC, all clients except client 1 are connected and PF policy will deliver the remaining data of them. Thus PF policy delivers a total amount of $\frac{MCR}{N+1} + NC$ data. The competitive ratio of PF policy cannot be better than:

$$\begin{aligned}\beta &= \frac{MC + NC}{\frac{MCR}{N+1} + NC} = \frac{\frac{N}{R} + N}{\frac{N}{N+1} + N} \\ &\to 1 + \frac{1}{R}\end{aligned}$$

as $N \to \infty$. □

2.8 Performance Comparison

For a large R, $\frac{e^{1/R}}{R(e^{1/R}-1)}$ is very close to $1 + \frac{1}{R}$, and it may seem that the competitive ratio of RR, MW, and PF is close to that of PD. However, we should interpret our results on competitive ratios as follows: Suppose it is required that online policies need to offload at least $\frac{1}{\beta}$ as much data as the optimal offline policy with unit capacity, for some given $\beta > 1$. With PD, the system needs to increase its capacity by approximately $\frac{1}{2(\beta-1)}$ times. On the other hand, even with the simple On-Off channels, the RR, MW, and PF policies still need at least $\frac{1}{\beta-1}$ capacity to achieve the requirement. In other words,PD needs about half as much capacity to provide the same guarantee as the other three policies. Thus, our policy is much more preferable to provide stringent performance guarantees. Fig 2.3 illustrates the capacity requirements for different β. In order to guarantee to offload at least 95% as much data as the optimal offline policy, PD needs to increase the capacity by 9.7 times, while the other policies needs to increase the capacity by at least 19 times.

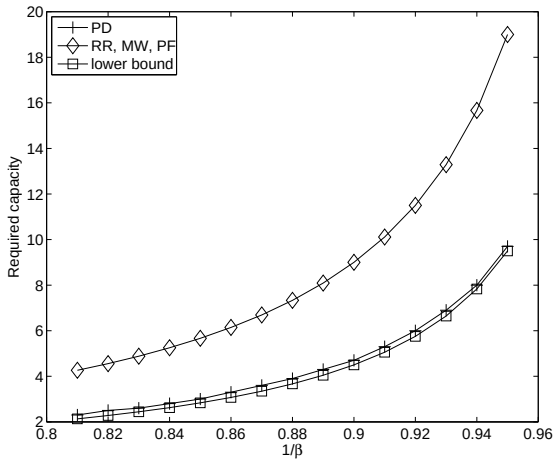

Figure 2.3: Capacity requirements of different policies.
Reprint with permission from [10].

2.9 Practical Implication

In this section, we will discuss the major practical implications of our work.

First, our work can be used for system planning. When a wireless operator is to deploy public WiFi APs, it can use the macro-scale statistics of system history to estimate the amount of resource needed for the offline policy to offload a certain amount of data.

For example, suppose we have collected the mobility patterns of all clients in a specific area and the clients' demand of the past 30 days. We use K_{imt} as clients' connection status to indicate the mobility pattern. We use C_i to represent the demand. Suppose the wireless operator needs to offload a total of C_0 amount of data. Here $C_0 \leq \sum_i C_i$. With the above information, the wireless operator can then derive the minimum required resource R_0 by simply solving the following linear programming problem.

$$
\begin{aligned}
&Min R_0 \\
&s.t. \sum_{imt} X_{imt} K_{imt} \geq C_0, \forall i \in \mathcal{I}, \forall m \in \mathcal{M}, t, \\
&\sum_{mt} X_{imt} K_{imt} \leq C_i, \forall i \in \mathcal{I}, \\
&\sum_{i} X_{imt} \leq R_0, \forall m \in \mathcal{M}, t, \\
&X_{imt} \geq 0, \forall i \in \mathcal{I}, m \in \mathcal{M}, t.
\end{aligned}
$$

Assuming the statistics of user mobility and demand do not change too much, the wireless operator is able to estimate the average total resource requirement with the past information. However, these macro-scale statistics become much less useful when it comes to online packet scheduling. For example, when two clients enter a coffee shop with WiFi at the same time, it is very difficult to predict which of them will leave the coffee shop first. In this case, it is not guaranteed to offload C_0 amount of data with the given R_0 resource. Our derivation of performance bound is indeed based on this difficulty. On the other hand, it is possible to formulate the scheduling problem as a Markov decision process (MDP) problem using past statistics [8]. However, this formulation typically requires each AP to solve a high-dimensional optimization problem for every packet transmission, and incurs preventively high computation complexity. Therefore, in many practical scenarios, simple online policies that do not rely on past statistics are needed.

Now, let's say that the wireless operator estimates that the offline policy needs an amount of R_0 resource to offload the desirable amount C_0 of traffic. How much resource does a simple online policy need to guarantee offloading $0.95C_0$ traffic?

Theorem 5 reveals that the answer is $9.5R_0$.

The second important implication concerns the comparison against other poplar policies including round robin, max-weight, and proportional fair policies. Our study reveals that our proposed policy only needs 50% as much resource as the other three to provide the same degree of performance guarantees. In other words, in view of competitive ratio, implementing our policy is as effective as doubling capacity. It is well known that 2X2 MIMO has the potential of doubling transmission rate. Therefore, our policy offers the same performance improvement as implementing 2X2 MIMO.

2.10 Simulation

In this section, we evaluate the performance of the two algorithms we proposed as well as that of the other three policies discussed in Secition 2.8.

We construct a system with 9 APs (3 by 3 grid). Each AP is 1000 meters away from its nearest neighbor , and has a transmission range of 400 meters. There are 200 clients which are divided into two groups: The first group is 100 stationary clients that are uniformly distributed within the coverage area of APs. The second group is 100 mobile users whose locations are chosen uniformly at random at each time t. In each group, the i-th client has $C_i = 100$, and $T_i = 50 + 50i$, if $i \leq 95$, and $C_i = 10,000$, $T_i = 5000(i - 95)$ if $95 < i \leq 100$. Fig. 4 shows a snapshot of the locations of all clients, where the circles represent the coverage area of APs.

The channel gain is determined by both pathloss and Rayleigh fading. The pathloss factor between an AP and a client is computed by $min\{1, 1/(distance/80)^2\}$. The Rayleigh fading factor is computed as $\sqrt{a^2 + b^2}$, where both a and b are Normal random variables with mean 0 and variance 1. Finally, the channel gain is the product of these two factors.

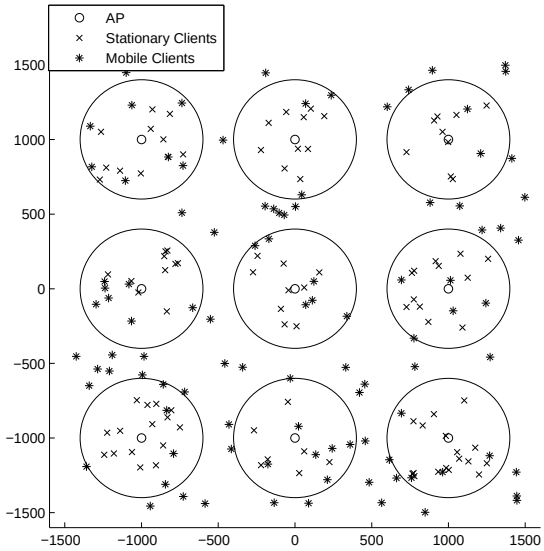

Figure 2.4: Location of APs and clients.
Reprint with permission from [10].

We consider both On-Off channels and general channels. With On-Off channels, we consider the channel to be ON if the channel gain is larger than $1/25$, in which case we set $K_{imt} = 1$. The threshold of $1/25$ is chosen as the pathloss factor when the distance is 400. With general channels, we set K_{imt} to be the channel gain. For each simulation run, we compute the portion of data that each policy is able to offload through WiFi. All simulation results are the average of 5 simulation runs.

The simulation results for both channels are shown in Figure 2.5 and Figure 2.6, respectively. The standard deviations of our algorithms are on the order of 10^{-4}, which shows that the deviation of our algorithm is very small. We notice that PD and LPF have almost identical performance. Recall that LPF is designed to be an approximation to PD with smaller overhead and easier implementation. These simulation results confirm that it is indeed an accurate approximation.

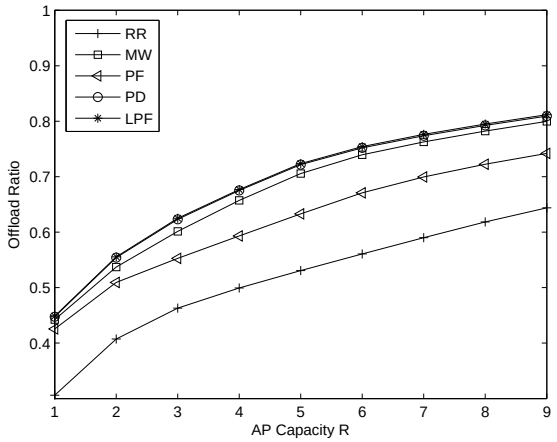

Figure 2.5: Performance comparison for On-Off channels.
Reprint with permission from [10].

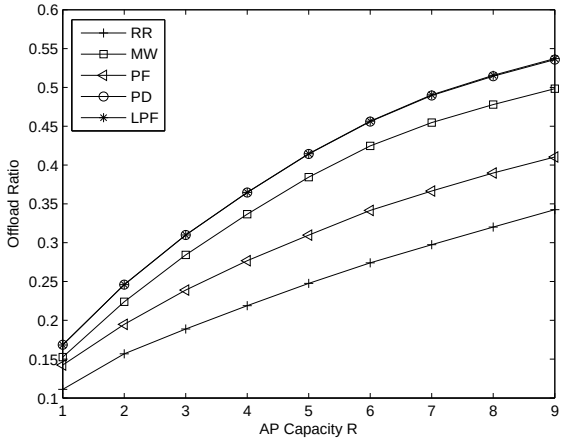

Figure 2.6: Performance comparison for general channels.
Reprint with permission from [10].

Further, we note that our policies outperform the other three policies in both scenarios. The theoretical analysis in Section 2.6 only proves that the worst-case performance of our policies is better than that of the other policies. These simulation results further suggest that our policies are still more preferable on average.

With On-Off channels, we notice that the data offload with our policies at R are no less than that RR offload at $2R$. For example, when $R = 2$, our policies are able to offload more data than RR when $R = 4$. The difference between our policies and the RR policy becomes even larger when general channels are considered. This is because the round robin policy does not consider channel capacity, and will use a large amount of time serving clients with poor channel qualities. For both On-Off channels and general channels, our policies outperform Max-Weight policy. Proportional fair policy does not have a good performance because it tries to even out each client's portion of received data. When clients have different data request and mobility pattern, the policy does not optimize the whole system's offload data amount.

2.11 Conclusion

In this section, we study the delayed mobile offloading problem with unpredictable user movement pattern. We aim to download as much data through WiFi as possible. We present two online algorithms for the problem and study their performance by comparing how much data they are able to offload to the optimal offline policy. First, we propose PD policy and prove that it is approximately $(R, \frac{1}{2R})$-competitive and achieves the optimal trade-off between capacity and amount of offloaded data. Second, we propose an alternative LPF policy that is easier to implement and has almost identical performance as PD. We also provide that the tight bound of online policies is $(R, \frac{1}{2R})$. Our policies are compared

with three commonly-used policies, including Round Robin, Max-Weight, and Proportional Fair policy, and and we prove that our policies only need half as much capacity to provide the same degree of performance guarantees under any mobility patterns. We simulate our proposed policies as well as the three commonly used policies to compare their performance in a randomly generated system. The results show that the proposed two policies have higher offloading ratio than the others.

3. ONLINE JOB ALLOCATION WITH HARD ALLOCATION RATIO REQUIREMENT *

3.1 Problem Overview

In this section, we discuss the problem of online job allocation for cloud computingwhere each job can only be served by a subset of servers [30]. Such a problem exists in many emerging Internet services, such as YouTube, Netflix, etc. For example, in the case of YouTube, each video is replicated only in a small number of servers, and each server can only serve a limited number of streams simultaneously. When a user accesses YouTube and makes a request to watch a video, this request needs to be allocated to one of the servers that not only stores the video but also has remaining capacity to process this request. If no server can process this request, the request is dropped, which can significantly impact user satisfaction.

There are many studies on obtaining statistics about user behaviors, video popularity, and access locality [31–35]. These studies provide important insights about system planning. However, they are not sufficient for job allocation. It is usually difficult to predict which video will go viral and generate most requests. Therefore, online job allocation policy that makes decisions solely based on current system state is needed.

The problem of online job allocation has attracted much attention. Most current studies study the performance of online policies by comparing the number of allocated jobs under online policies against that of the an offline policy with full knowledge about all future arrivals. For example, Karp, Vazirani and Vazirani [36] propose an online policy that is guaranteed $1 - \frac{1}{e} \approx 63\%$ of jobs, given that the

*Reprinted with permission from "Online job allocation with hard allocation ratio requirement" by Han Deng, I-Hong Hou, © 2016, IEEE INFOCOM. [30]

offline policy allocates all jobs. They also show that no online policy can guarantee to allocate more jobs than their proposed policy.

These studies are still insufficient for many practical scenarios. In particular, one can argue that most practical applications demand a much higher ratio of allocated jobs than 63%. In order to meet this demand, current practice is to add redundancy and increase the capacity of data centers to accommodate unpredictable patterns of job arrivals. Motivated by this observation, we seek to address the following question: How much capacity is needed to guarantee the allocation of, say, 95% of jobs?

To answer this problem, we first formulate a linear programming problem for job allocation. The uncertainty in future job arrivals corresponds to the uncertainty in some of the parameters. Also, increasing the capacity of the servers corresponds to relaxing some of the constraints in the linear programming problem.

Using this formulation, we propose two simple online scheduling policies and derive closed-form expressions for their performance. Specifically, we prove that, in order to allocate at least $1 - \frac{1}{\theta}$ of jobs, the two policies only need to increase the capacity by $\ln \theta$ times. We also prove that no online policy can guarantee to allocate the same ratio of jobs with less capacity, and hence our policies are optimal.

We further evaluate two popular online job allocation policies. Surprisingly, we prove that, to guarantee to allocate at least $1 - \frac{1}{\theta}$, these two policies require at least $\theta - 1$ times capacity. Therefore, they are both an order worse than our policies.

The allocation ratio guarantees stated above need to hold for every sample path. We also consider the scenario where jobs arrivals are generated by some unknown random process, and one only needs guarantees on the expected value of allocation ratio. By both theoretical analysis and numerical studies, we demonstrate that our policies remain much better than the other two policies for this scenario.

Next, we consider the setting where jobs arrive at arbitrary time in a slotted time system. Each job has a deadline which is revealed upon arrival. We propose an online policy and study its performance.

3.2 Related Work

There are many studies on YouTube videos about their statistical properties [35,37,38]. Studies on online learning further investigate the possibility to predict the future video requests. The problem of “learn from expert advise" was first studied by Littlestone and Warmuth [39], DeSantis, Markowsky, and Wegman [40]. Later “learn from examples" was studied. The Winnow algorithm was proposed and studied by Littlestone and Nicholas [41], [42]. The algorithm applies well to practical tasks such as on World Wide Web [43]. Other sequence prediciton research also include the studies by Nicolo and Gabor [44], Hutter [45].

The online job allocation problem is an online matching procedure which aim to make the best decision on job-server pair to maximize the number of jobs get matched. The problem of online bipartite matching was studied by Karp, Vazirani, and Vazirani [36]. They use an adversary model and studied GREEDY which achieves a matching ratio of $1/2$ and RANKING which achieves $1 - 1/e$. They further showed that no algorithm can achieve a better ratio than $1 - 1/e$. Other models, which are based on further assumption on arrival pattern, have also been studied. Random arrival model has been studied by Goel and Mehta [46], and Karande, Mehta and Tripathi [47]. They show that GREEDY achieves a matching ratio of $1 - 1/e$ and RANKING achieves greater than $1 - 1/e$. Known distribution model was introduced by Feldman, Mehta, Mirrokni, and Muthukrishnan [48]. They provide a two-suggested-matching algorithm which achieves a ratio of 67%. Kalyanasundaram and Pruhs studied the online b-matching problem [49] which

can be seen as the job allocation problem with server capacity b. They presented BALANCE algorithm and proved that it approaches $1 - 1/e$. Applications of on-line matching to ad-words problem, which is an allocation of bidders to key words within the budget limit of each bidder, have been studied in [46], [50].

3.3 System Model

We consider a system with multiple non-identical servers. Jobs arrive at the system sequentially. Jobs are of different types, and each job can only be served by a subset of the servers. For example, in the application of on-demand video streaming, a job is a request for one video, and can therefore only be served by servers that possess the video. We assume that when a job enters the system, it needs to be allocated to a server immediately. Jobs that cannot be allocated upon arrivals are discarded from the system. We also assume that jobs cannot be moved once they are allocated to servers, as moving jobs between servers cause additional costs on job migration. A similar model has been used in [51].

We use $\mathcal{J}$ to denote the set of servers, and $\mathcal{I} = \{1, 2, \dots\}$ to denote the arrival sequence of jobs. Each job i can be served by a subset $K_i \subseteq \mathcal{J}$ of servers. Upon its arrival, job i reveals its K_i, and the system either allocates it to a server or discards it. Each job takes one unit of capacity in the server to which the job is allocated to. A server j has a total amount of C_j units capacity, and can therefore at most serve C_j jobs. We use X_{ij} to denote the assignment of the jobs. If $X_{ij} = 1$, then job i is assigned to server j. If $X_{ij} = 0$, job i is not assigned to server j.

We aim to maximize the number of jobs that can be served. There are two constraints: first, the total job allocated to any server should be less or equal to its capacity; second, each job can be allocated to at most one server. The problem of maximizing the number of served jobs can be formulated as the following linear

programming problem:

Allocation:

$$Max \sum_{ij} X_{ij} \tag{3.1}$$

$$s.t. \sum_{i:j \in K_i} X_{ij} \leq C_j, \forall j \in \mathcal{J}, \tag{3.2}$$

$$\sum_{j:j \in K_i} X_{ij} \leq 1, \forall i \in \mathcal{I}, \tag{3.3}$$

$$X_{ij} \geq 0, \forall i \in \mathcal{I}, j \in \mathcal{J}. \tag{3.4}$$

Since $X_{ij} = 1$ if job i is served by j, (3.1) is the total number of served jobs. On the other hand, (3.2) states that each server j can at most serve C_j jobs, and (3.3) states that each job can be served by at most one server. In this formulation, we allow X_{ij} to be any real number between 0 and 1, while X_{ij} needs to be either 0 or 1 according to our model. Therefore, **Allocation** describes an upper-bound of the number of jobs that can be allocated.

Solving **Allocation** is straightforward when one has knowledge of all its parameters $\{K_i\}$ and $\{C_j\}$. However, as jobs arrive sequentially, the system needs to make allocation decisions without knowledge of future jobs. We say that an allocation policy is an *online policy* if it makes allocation decisions only based on jobs that have already arrived. On the other hand, an allocation policy is an *offline policy* if it has full knowledge about all future job arrivals, and can therefore find the optimal solution to **Allocation**.

We consider that the service provider can increase server capacity to allocate more jobs. When the system capacity is increased by R times, a server j with C_j capacity will have RC_j capacity after the increase. We can now formulate the

following linear programming problem:

Allocation(R):

$$Max \sum_{ij} X_{ij} \tag{3.5}$$

$$s.t. \sum_{i:j\in K_i} X_{ij} \leq RC_j, \forall j \in \mathcal{J}, \tag{3.6}$$

$$\sum_{j:j\in K_i} X_{ij} \leq 1, \forall i \in \mathcal{I}, \tag{3.7}$$

$$X_{ij} \geq 0, \forall i \in \mathcal{I}, j \in \mathcal{J}. \tag{3.8}$$

We evaluate the performance of online policies by comparing the number of allocated jobs under online policies with R times capacity against that under offline policy with unit capacity. Specifically, given $\{K_i\}$ and $\{C_j\}$, let Γ_{opt} be the optimal value of $\sum_{ij} X_{ij}$ in **Allocation**, and $\Gamma_\eta(R)$ be the value of $\sum_{ij} X_{ij}$ in **Allocation(R)** under policy η. We define the *competitive ratio per sample path* as follows:

Definition 2. *An online policy η is said to be* (R, θ)-competitive-per-sample-path *if* $\Gamma_{opt}/\Gamma_\eta(R) \leq \theta$*, for all* $\{K_i\}$ *and* $\{C_j\}$ *with* $C_j \geq C_{min}$*, as* $C_{min} \to \infty$.

Definition 2 defines competitive ratio based on the worst-case sample path. This definition may ignore effects of statistic multiplexing. In practice, jobs may arrive according to some random process. Therefore, the arrivals of different types of jobs are likely to be interwined.

We can expand our model to accommodate the random nature of job arrivals. Given $\{K_i\}$, we can consider the case where the actual arrival sequence is a random permutation of $\mathcal{I} = \{1, 2, \dots\}$. Let $E[\Gamma_\eta(R)]$ be the expected number of allocated jobs under η with R times capacity when the arrival sequence is a random permutation. We then define the *expected competitive ratio* as follows

Definition 3. *An online policy η is said to be* (R, θ)-competitive-in-expectation *if* $\Gamma_{opt}/E[\Gamma_\eta(R)] \leq \theta$*, for all* $\{K_i\}$ *and* $\{C_j\}$ *with* $C_j \geq C_{min}$*, as* $C_{min} \to \infty$.

It is obvious that the competitive ratio per sample path cannot be better than the expected competitive ratio.

Lemma 3. *A* (R, θ)*-competitive-per-sample-path policy is* (R, θ)*-competitive-in-expectation.*

3.4 Two Online Allocation Policies and Their Competitive Ratios

In this section, we propose online policies and analyze their competitive ratios. Our analysis is based on the Weak Duality Theorem of linear programming [52]. The dual problem of **Allocation** can be written as:

Dual:

$$Min \sum_j C_j \alpha_j + \sum_i \beta_i, \tag{3.9}$$

$$s.t.\ \alpha_j + \beta_i \geq 1, \forall i \in \mathcal{I}, j \in K_i \tag{3.10}$$

$$\alpha_j \geq 0, \forall j, \tag{3.11}$$

$$\beta_i \geq 0, \forall i, \tag{3.12}$$

where each α_j corresponds to a constraint in (3.2), and each β_i corresponds to a constraint in (3.3). The following lemma is then a direct result of the Weak Duality Theorem.

Lemma 4. *Given any vectors of* $\{\alpha_j\}$ *and* $\{\beta_i\}$ *that satisfy the constraints (4.10)–(4.12), we have* $\sum_j C_j \alpha_j + \sum_i \beta_i \geq \Gamma_{opt}$.

We now introduce an online policy. This policy maintains a variable α_j for each server j. When the system starts, it sets $\alpha_j \equiv 0$ initially. When a job i arrives,

the policy checks the values of α_j for all $j \in K_i$, and selects j^* as the one with the minimum value of α_j. If $\alpha_{j^*} < 1$, job i is assigned to server j^*, and therefore $X_{ij^*} = 1$. The value of α_{j^*} is updated to be $\alpha_{j^*}(1 + \frac{1}{C_{j^*}}) + \frac{1}{(d_{j^*} - 1)C_{j^*}}$, where we set $d_j = (1 + 1/C_j)^{RC_j}$, for all j. The value of d_j is chosen to achieve the optimal competitive ratio, as will be shown in the proof of Lemma 5. On the other hand, if $\alpha_{j^*} \geq 1$, job i is discarded. The complete policy is described in Algorithm 4.

Algorithm 4 PD Algorithm

Initially, $\alpha_j = 0$, $\beta_i = 0$, $X_{ij} = 0$.
$d_j \leftarrow (1 + 1/C_j)^{RC_j}, \forall j$.
for each arriving job i **do**
  $j^* \leftarrow \text{argmin}_{j \in K_i} \alpha_j$.
  if $\alpha_{j^*} < 1$ **then**
    $\beta_i \leftarrow 1 - \alpha_{j^*}$.
    $\alpha_{j^*} \leftarrow \alpha_{j^*}(1 + \frac{1}{C_{j^*}}) + \frac{1}{(d_{j^*} - 1)C_{j^*}}$.
    $X_{ij^*} \leftarrow 1$.
    Job i is assigned to server j^*.
  else
    Discard job i.
  end if
end for

We first need to show that the vector $\{X_{ij}\}$ produced by this policy satisfies all constraints of **Allocation(**R**)**, so that the policy never assigns a job to a server that is already fully utilized.

Lemma 5. *Let $\alpha_j[n]$ be the value of α_j after n jobs have been allocated to server j.*

Then,

$$\alpha_j[n] = (\frac{1}{d_j - 1})(d_j^{n/RC_j} - 1). \tag{3.13}$$

Proof. Here we prove (3.13) by induction.

Initially, when $n = 0$, $\alpha_j[0] = 0 = (\frac{1}{d_j - 1})(d_j^0 - 1)$ and (3.13) holds.

Suppose (3.13) holds when $n = k$. When the $(k + 1)$-th job is allocated server j, we have

$$\begin{aligned}
\alpha_j[k+1] =& \alpha_j[k](1 + \frac{1}{C_j}) + \frac{1}{(d_j - 1)C_j} \\
=& \frac{1}{(d_j - 1)}(d_j^{k/RC_j} - 1)(1 + \frac{1}{C_j}) + \frac{1}{(d_j - 1)C_j} \\
=& \frac{1}{(d_j - 1)}[d_j^{(k+1)/RC_j} - 1],
\end{aligned}$$

and (3.13) still holds for $n = k + 1$. By induction, (3.13) holds for all n. □

With Lemma 5, $\alpha_j = 1$ when RC_j jobs have been allocated to server j. Since Algorithm 4 only allocates jobs to servers with $\alpha_j < 1$, our policy does not violate any constraints in **Allocation(**R**)**.

Next, we study the competitive ratio of Algorithm 4.

Theorem 10. *Algorithm 4 is* $(R, \frac{e^R}{e^R - 1})$*-competitive-per-sample-path.*

Proof. We prove Theorem 10 by three steps:

First, we show that solutions $\{\alpha_j\}$ and $\{\beta_i\}$ satisfy all constraints in **Dual**. Initially, α_j and β_i are set to be 0. By step 7 in Algorithm 4, α_j is non-decreasing throughout the execution of the policy, and hence (4.11) holds. Also, by Lemma 5,

$\alpha_j \leq 1$, for all j. When a job i arrives, our policy sets $j^* \leftarrow \operatorname{argmin}_{j \in K_i}\{\alpha_j\}$. If $\alpha_{j^*} = 1$, we have $\alpha_j = 1$ for all $j \in K_i$ and $\beta_i = 0$. Hence, both constraints (4.10) and (4.12) hold. On the other hand, if $\alpha_{j^*} < 1$, $\beta_i = 1 - \alpha_{j^*} \geq 1 - \alpha_j$, for all $j \in K_i$. Both constraints (4.10) and (4.12) still hold.

Next, we derive the ratio between $\sum_{ij} X_{ij}$ and $\sum_j C_j \alpha_j + \sum_i \beta_i$. Both formulas are initially 0. We now consider the amounts of change of these two formulas when a job i arrives. We use $\Delta P(R)$ to denote the change of $\sum_{ij} X_{ij}$, and ΔD to denote the change of $\sum_j C_j \alpha_j + \sum_i \beta_i$.

If job i is discarded, then $\{X_{ij}\}$, $\{\alpha_j\}$ and $\{\beta_i\}$ remain unchanged, and therefore $\Delta P(R) = \Delta D = 0$.

On the other hand, consider the case when job i is assigned to server j. We have $X_{ij} = 1$ and $\Delta P(R) = 1$. We also have

$$\begin{aligned}\frac{\Delta D}{\Delta P(R)} = \Delta D =& C_j\left(\frac{\alpha_j}{C_j} + \frac{1}{(d_j - 1)C_j}\right) + 1 - \alpha_j \\ =& 1 + \frac{1}{d_j - 1} = \frac{d_j}{d_j - 1} \\ =& \frac{(1 + 1/C_j)^{RC_j}}{(1 + 1/C_j)^{RC_j} - 1}.\end{aligned}$$

When we imposes a lower bound on C_j by requiring $C_j \geq C_{min}$, for all j, and let $C_{min} \to \infty$, we have $\frac{\Delta D}{\Delta P(R)} \to \frac{e^R}{e^R - 1}$, whenever a job i is allocated to some server. Therefore, we have, under Algorithm 4,

$$\frac{\sum_j C_j \alpha_j + \sum_i \beta_i}{\sum_{ij} X_{ij}} = \frac{e^R}{e^R - 1}. \tag{3.14}$$

Finally, by Lemma 10, we establish that Algorithm 4 is $(R, \frac{e^R}{e^R - 1})$-competitive-per-sample-path. □

Algorithm 4 relies on the usage of artificial variables $\{\alpha_j\}$ and $\{d_j\}$. Below, we introduce a second online policy that not only is simpler, but also conveys better intuition. The policy is called Join-Least-Utilization (JLU) policy. When a job arrives, JLU simply allocates the job to the server with the smallest utilization ratio, which is the number of allocated jobs at a server divided by its capacity. Specifically, let n_j be the number of jobs that have already been allocated to server j. When job i arrives, it is allocated to $\operatorname{argmin}_{j \in K_i} \frac{n_j}{RC_j}$.

The complete policy is described in Algorithm 5. While the algorithm still involves $\{\alpha_j\}$, $\{d_j\}$, and $\{\beta_i\}$, these variables are introduced solely for the purpose of the analysis of competitive ratio. They can be omitted in actual implementation.

Algorithm 5 JLU

Initially, $\alpha_j = 0$, $\beta_i = 0$, $X_{ij} = 0$, $n_j = 0$.
$d_j \leftarrow (1 + 1/C_j)^{RC_j}, \forall j$.
for each arriving job i **do**
  $j^* \leftarrow \operatorname{argmin}_{j \in K_i} \{\frac{n_j}{RC_j}\}$.
  if $n_{j^*} < RC_{j^*}$ **then**
    $X_{ij^*} \leftarrow 1$.
    $\alpha_{j^*} \leftarrow \alpha_{j^*}(1 + \frac{1}{C_{j^*}}) + \frac{1}{(d_{j^*} - 1)C_{j^*}}$.
    $n_{j^*} \leftarrow n_{j^*} + 1$.
    Job i is assigned to server j^*.
    $\beta_i \leftarrow 1 - \min_{j \in K_i} \alpha_j$.
  else
    Discard job i.
  end if
end for

Lemma 6. *For any $\delta > 0$, there exists a finite C_{min} such that, by requiring $C_j > C_{min}$,*

for all j, we have

$$\alpha_{j_1} - \alpha_{j_2} > \delta \Longrightarrow \frac{n_{j_1}}{RC_{j_1}} > \frac{n_{j_2}}{RC_{j_2}}, \tag{3.15}$$

for all $j_1, j_2 \in \mathcal{J}$, throughout the execution of JLU.

Proof. The equation for updating α_j in JLU is the same as that in Algorithm 4. By Lemma 5, at any point of time, we have

$$\alpha_j = (\frac{1}{d_j - 1})(d_j^{n_j/RC_j} - 1) \tag{3.16}$$

$$= \frac{1}{(1 + 1/C_j)^{RC_j} - 1}[(1 + 1/C_j)^{n_j} - 1], \tag{3.17}$$

as $d_j = (1 + 1/C_j)^{RC_j}$.

Note that $\alpha_j \to \frac{e^{n_j/C_j} - 1}{e^R - 1}$ for a fixed R and all $\frac{n_j}{C_j} \leq R$, as $C_j \to \infty$. Thus, for any $\delta > 0$, there exist a finite C_{min} such that, by requiring $C_j > C_{min}$ for all j, we have $|\alpha_j - \frac{e^{n_j/C_j} - 1}{e^R - 1}| < \delta/2$ for all j. Therefore, for any two servers j_1 and j_2, we have

$$|(\alpha_{j_1} - \alpha_{j_2}) - (\frac{e^{n_{j_1}/C_{j_1}} - 1}{e^R - 1} - \frac{e^{n_{j_2}/C_{j_2}} - 1}{e^R - 1})| \tag{3.18}$$

$$< |\alpha_{j_1} - \frac{e^{n_{j_1}/C_{j_1}} - 1}{e^R - 1}| + |\alpha_{j_2} - \frac{e^{n_{j_2}/C_{j_2}} - 1}{e^R - 1}| < \delta. \tag{3.19}$$

This implies that

$$\alpha_{j_1} - \alpha_{j_2} > \delta \Longrightarrow \frac{e^{n_{j_1}/C_{j_1}} - 1}{e^R - 1} > \frac{e^{n_{j_2}/C_{j_2}} - 1}{e^R - 1} \tag{3.20}$$

$$\Longrightarrow \frac{n_{j_1}}{RC_{j_1}} > \frac{n_{j_2}}{RC_{j_2}}, \tag{3.21}$$

and the proof is complete.

□

Theorem 11. *JLU is $(R, \frac{e^R}{e^R - 1})$-competitive-per-sample-path.*

Proof. The proof is very similar to that of Theorem 10.

By Lemma 5, $\frac{n_j}{RC_j} < 1 \Leftrightarrow \alpha_j < 1$. Therefore, under JLU, an arriving job i is allocated if and only if $\min_{j \in K_i} \alpha_j < 1$. For any $\delta > 0$, we pick a sufficiently large C_{min} so that (3.15) holds.

We can establish that the solutions $\{X_{ij}\}$, $\{\alpha_j\}$, and $\{\beta_i\}$ produced by JLU satisfy all constraints in **Allocation(R)** and **Dual** using an argument that is virtually the same as that in the proof of Theorem 10.

Next, we derive the ratio between $\sum_{ij} X_{ij}$ and $\sum_j C_j \alpha_j + \sum_i \beta_i$. Both formulas are initially 0. We now consider the amounts of change of these two formulas when a job i arrives. We use $\Delta P(R)$ to denote the change of $\sum_{ij} X_{ij}$, and ΔD to denote the change of $\sum_j C_j \alpha_j + \sum_i \beta_i$.

If job i is discarded, then $\Delta P(R) = \Delta D = 0$.

On the other hand, consider the case when job i is assigned to server $j^* = \operatorname{argmin}_{j \in K_i} \frac{n_j}{RC_j}$. By Lemma 6, $\alpha_j \geq \alpha_{j^*} - \delta$, for all $j \in K_i$.

We now have

$$\begin{aligned}
&\frac{\Delta D}{\Delta P(R)} = \Delta D \\
=&C_{j^*}(\frac{\alpha_{j^*}}{C_{j^*}} + \frac{1}{(d_{j^*}-1)C_{j^*}}) + 1 - \min_{j \in K_i} \alpha_j \\
\leq&C_{j^*}(\frac{\alpha_{j^*}}{C_{j^*}} + \frac{1}{(d_{j^*}-1)C_{j^*}}) + 1 - \alpha_{j^*} + \delta \\
=&\frac{(1+1/C_{j^*})^{RC_{j^*}}}{(1+1/C_{j^*})^{RC_{j^*}} - 1} + \delta.
\end{aligned}$$

Let $C_{min} \to \infty$, and we have under JLU,

$$\frac{\sum_j C_j \alpha_j + \sum_i \beta_i}{\sum_{ij} X_{ij}} \leq \frac{e^R}{e^R - 1} + \delta, \tag{3.22}$$

for any $\delta > 0$

Finally, by Lemma 10, we establish that JLU is $(R, \frac{e^R}{e^R-1})$-competitive-per-sample-path. □

By Lemma 3, we also have the following theorem.

Theorem 12. *Both Algorithm 4 and JLU are* $(R, \frac{e^R}{e^R-1})$*-competitive-in-expectation.*

3.5 Lower Bounds of Competitive Ratios

In Section 3.4, we show that our online policies are both $(R, \frac{e^R}{e^R-1})$-competitive-per-sample-path. In this section, we will study the lower bound for the competitive ratio per sample path. We focus on a special class of systems described below:

A system in this class has N servers with capacity C each, where C is chosen to be a multiple of $N!$. A total number of NC jobs arrive in sequence, and they are separated into N groups, where the k-th group contains jobs $\{(k-1)C+1, (k-1)C+2, \ldots, kC\}$. Jobs in the same group can be served by the same subset of servers. The subset of servers that can serve a job i, i.e., K_i is constructed as follows: Jobs in the first group $\{1, 2, \ldots, C\}$ can be served by all servers, i.e., $K_i = \mathcal{J}$. For each job in the $(k+1)$-th group, its K_i is obtained by removing one element from that for jobs in the k-th group. More specifically, for a job i_1 in the k-th group and a job i_2 in the $(k+1)$-th group, we have $K_{i_2} \subsetneq K_{i_1}$ and $|K_{i_2}| = |K_{i_1}| - 1$. It is easy to verify that an offline policy can allocate all NC jobs for all systems in this class.

We consider a policy, namely, EVEN, that evenly distributes jobs in the same group among all available servers. We first establish the number of jobs that can be allocated by EVEN, and then show that no policy can guarantee to allocate more jobs than EVEN within this class of systems.

Lemma 7. *When the capacity is increased by R times, EVEN serves at most $(N - \frac{N+1}{e^R} + 2)C$ jobs.*

Proof. Since EVEN allocates jobs evenly on all available servers, each server gets $\frac{C}{N}$ jobs in the first group of C jobs. Similarly, as jobs in the k-th group can be served by $N - k + 1$ servers, each server that can serve this group gets $\frac{C}{N-k+1}$ jobs in this group, unless the server is already fully utilized.

Consider the case when each server has RC capacity. Suppose the system can only serve up to the $(k+1)$-th group, that is, servers are still not fully utilized after serving the k-th group. We then have

$$\begin{aligned}
&C(\frac{1}{N} + \frac{1}{N-1} + \frac{1}{N-2} + \ldots + \frac{1}{N-k+1}) < RC \\
\Rightarrow &\int_{N-k+1}^{N+1} \frac{1}{x} dx < (\frac{1}{N} + \frac{1}{N-1} + \ldots + \frac{1}{N-k+1}) < R \\
\Rightarrow &\log(N+1) - \log(N-k+1) = \log \frac{N+1}{N-k+1} < R \\
\Rightarrow &k < N + 1 - \frac{N+1}{e^R}
\end{aligned}$$

Since servers can serve up to the $(k+1)$-th group and become fully utilized after the arrival of this group, the number of jobs served in the system is then at most $(k+1)C < (N - \frac{N+1}{e^R} + 2)C$ □

Lemma 8. *When the parameters R, N, and C are fixed, no online policy can guarantee to allocate more jobs than EVEN.*

Proof. We consider an alternative policy ALT and show that it cannot allocate more jobs than EVEN. Given R, N, and C, we construct K_i iteratively as follows: The first group can be served by all servers. Let j_1 be the server with the least jobs. We then choose $K_i = \mathcal{J} \backslash \{j_1\}$ for the second group. Similarly, let j_k be the server

with the least jobs among severs that can serve the k-th group. We choose $K_i = \mathcal{J}\backslash\{j_1, j_2 \ldots, j_k\}$ for the $(k+1)$-th group. Under this arrival sequence, the total amount of unused capacity in all servers is at least $(RC - \frac{C}{N}) + (RC - \frac{C}{N} - \frac{C}{N-1}) + \ldots$, which is the total amount of unused capacity by EVEN. Therefore, ALT cannot allocate more jobs than EVEN. □

Theorem 13. *Any online policy cannot be better than* $(R, \frac{e^R}{e^R - 1})$*-competitive-per-sample-path.*

Proof. This is a direct result of Lemmas 7 and 8. □

3.6 Competitive Ratios of Other Widely Policies

In this section, we study the competitive ratios of two widely used polices.

3.6.1 Join the Shortest Queue

The first policy is the join the shortest queue (JSQ) policy, which allocates jobs to servers with the smallest number of jobs. Specifically, let n_j be the number of jobs that have already been allocated to server j. When a new job i arrives, it is allocated to $\text{argmin}_{j \in K_i}\{n_j | n_j < RC_j\}$, if there exists a server $j \in K_i$ with $n_j < RC_j$.

Theorem 14. *JSQ cannot be better than* $(R, 1 + \frac{1}{R})$*-competitive-per-sample-path.*

Proof. Given R, we construct a system with two types of servers, $J1$ and $J2$, and two types of jobs, $I1$ and $I2$. Type $I1$ jobs can be served by all servers, while type $I2$ jobs can only be served by type $J2$ severs.

The system is described as Fig 3.1. It has one type $J1$ server with capacity MC and K type $J2$ servers with capacity C. The job arrival sequence is as follows: first MC jobs of type $I1$ arrive; then KC jobs of type $I2$ arrive. The values of M and K are chosen such that $R = M/(K+1)$. The jobs (or servers) of same type are in

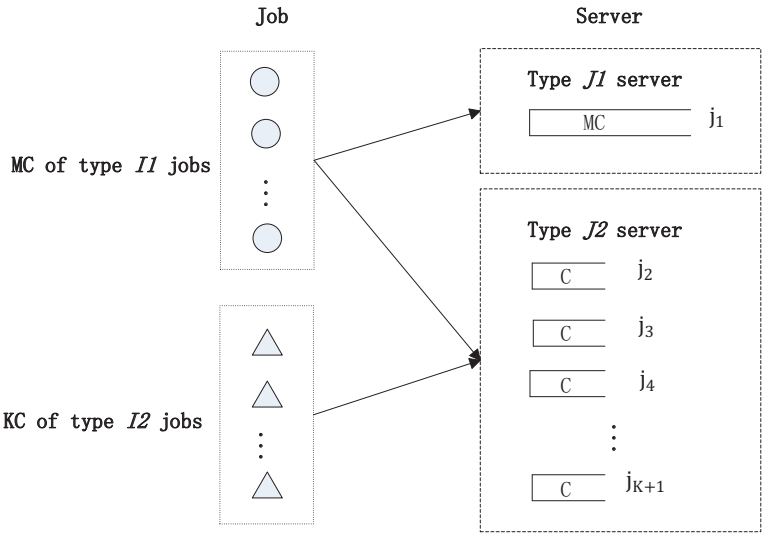

Figure 3.1: System illustration for the analysis of JSQ.
Reprint with permission from [30].

the same square box. An arrow line indicates that the job can be allocated to the server.

The optimal offline policy is to allocate all type $I1$ jobs to the server of type $J1$, and allocate all type $I2$ jobs to type $J2$ servers. This allocation can allocate all $MC + KC$ jobs, and $\Gamma_{opt} = MC + KC$.

Now, consider the performance of JSQ when the server capacity is increased by R times. After the increase, the type $J1$ server has RMC capacity, and all other servers have RC capacity. The first MC arrivals are all type $I1$ jobs, who can be served by all servers. Therefore, JSQ evenly distribute these jobs to all servers, and each server gets $MC/(K+1) = RC$ jobs. Next, type $I2$ jobs arrive, and they can only be served by type $J2$ servers. However, at this point, all type $J2$ servers are fully utilized, and no type $I2$ job can be served. The total number of served jobs under JSQ is $\Gamma_{JSQ}(R) = MC$.

We then have

$$\frac{\Gamma_{opt}}{\Gamma_{JSQ}(R)} = \frac{MC + KC}{MC} \tag{3.23}$$

$$= \frac{M + K}{M} \to 1 + \frac{1}{R}, \tag{3.24}$$

as $K \to \infty$, and $M = R(K + 1)$.

□

Theorem 15. *JSQ cannot be better than* $(R, 1 + \frac{1}{R^2+2R})$*-competitive-in-expectation.*

Proof. Given R, we use the same system as that in the proof of Theorem 14, but consider that the actual arrival sequence is a random permutation of all jobs. We still have $\Gamma_{opt} = MC + KC$. In this proof, we choose M and K such that $R = \frac{M}{K}$.

Now, consider the performance of JSQ when the server capacity is increased by R times. Type $J2$ servers can serve all jobs, while the type $J1$ server can only serve jobs of type $I1$. Under JSQ, whenever a type $J2$ server has the least jobs among all servers, the next job will be allocated to this server, regardless of the type of the job. Therefore, under JSQ, all type $J2$ servers will have at least one less job than the number of jobs at the $J1$ server before all type $J2$ servers are fully utilized. Hence, after $(K + 1)RC$ arrivals, all type $J2$ servers are fully utilized. From then on, only type $I1$ jobs can be served. Further, there are MC type $I1$ jobs and KC type $I2$ jobs. Under a random permutation, the average number of type $I1$ jobs that arrive after the first $(K+1)RC$ arrivals is $\frac{MC}{MC+KC}[MC+KC-(K+1)RC] = MC - \frac{M(K+1)RC}{M+K}$. The expected number of jobs served by JSQ is then

$$E[\Gamma_{JSQ}(R)] \leq (K + 1)RC + MC - \frac{M(K + 1)RC}{M + K} \tag{3.25}$$

$$= MC + (K + 1)RC\frac{K}{M + K}, \tag{3.26}$$

and hence

$$\begin{aligned}\frac{\Gamma_{opt}}{E[\Gamma_{JSQ}(R)]} &\geq \frac{MC+KC}{MC+(K+1)RC\frac{K}{M+K}} \\ &= \frac{RK+K}{RK+(K+1)RK/(RK+K)} \\ &\to \frac{(R+1)^2}{R(R+1)+R} \\ &= 1+\frac{1}{R^2+2R},\end{aligned}$$

as $K \to \infty$ and $M = KR$. □

3.6.2 Join the Most Residue Queue

Join the most residue queue (JMQ) allocates jobs to servers with most remaining space, which is the server capacity minus the number of allocated jobs in this server. Let RC_j be the capacity of server j, n_j be the number of jobs that have already been allocated to j. The arriving job i is allocated to server $\text{argmax}_{j\in K_i}\{RC_j - n_j | n_j < RC_j\}$, if there exists a server $j \in K_i$ with $n_j < RC_j$.

Theorem 16. *Join the most residue queue policy cannot be better than* $(R, 1+\frac{1}{R})$*-competitive-per-sample-path.*

Proof. Given R, we construct a system with two types of servers, $J1$ and $J2$, and two types of jobs, $I1$ and $I2$. Type $I1$ jobs can only be served by type $J1$ servers, while type $I2$ jobs can be served by all servers.

The system is described as Fig 3.2. The system has one type $J1$ server with capacity $(M+1)C$, and K type $J2$ servers with capacity C. The job arrival sequence is as follows: first KC jobs of type $I2$ arrive; then $(M+1)C$ jobs of type $I1$ arrive. The value of M and K are chose such that $R = K/M$.

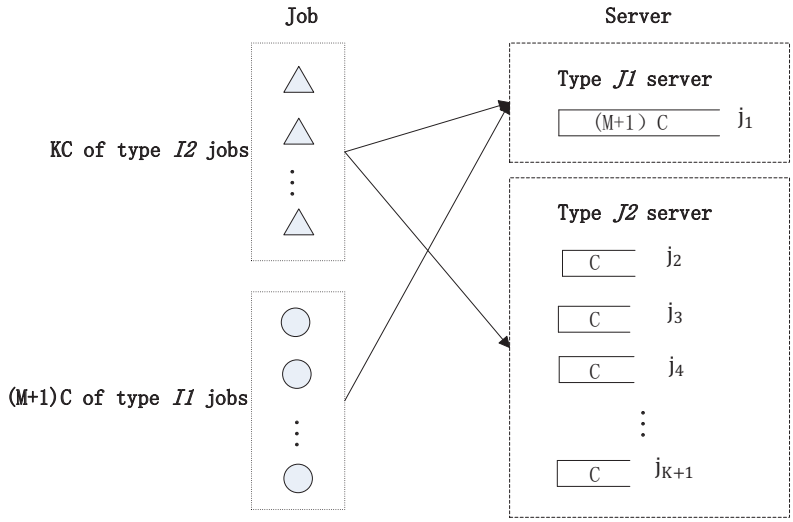

Figure 3.2: System illustration for the analysis of JMQ.
Reprint with permission from [30].

The optimal offline policy is to allocate all type $I2$ jobs to servers of type $J2$ and all type $I1$ jobs to the type $J1$ server. The total number of jobs allocated by this policy is $(M+1)C + KC$, and $\Gamma_{opt} = (M+1)C + KC$.

When the server capacity is increased by R times, type $J1$ server has $R(M+1)C$ capacity, and type $J2$ servers have RC capacity. The first arriving $KC = MRC$ jobs are of type $I2$. They can be served by both type $J1$ and $J2$ servers. JMQ allocates all these jobs to type $J1$ server. Type $J1$ server can therefore serve only RC jobs of type $I1$. The total number of jobs served is $R(M+1)C$.

We then have

$$\frac{\Gamma_{opt}}{\Gamma_{JMQ}(R)} = \frac{(M+1)C + KC}{R(M+1)C} \tag{3.27}$$

$$= \frac{M}{M+1} + \frac{1}{R} \to 1 + \frac{1}{R}, \tag{3.28}$$

as $M \to \infty$, and $K = MR$. □

Theorem 17. *JMQ cannot be better than* $(R, 1 + \frac{1}{R^2+2R})$*-competitive-in-expectation.*

Proof. We use the system in the proof of Theorem 16 but consider that the actual arrival sequence is a random permutation of all jobs. First, we have $\Gamma_{opt} = (M + 1)C + KC$. Now we study $E[\Gamma_{JMQ}(R)]$. In this proof, we choose M and K such that $K = MR - 1$.

The type $J1$ server can serve all jobs and has MRC more capacity than others. Thus, the first MRC jobs will be allocated to the type $J1$ server, regardless of job types. After the first MRC arrivals, the type $J1$ server has RC capacity left and hence at most RC more type $I1$ jobs can be served. Since there are $(M + 1)C$ type $I1$ jobs and KC type $I2$ jobs, the average number of type $I1$ jobs among the first MRC arrivals is $MRC \cdot \frac{M+1}{M+1+K}$. The expected number of allocated jobs is then no more than $RC + MRC\frac{M+1}{M+1+K} + KC$. Therefore we have:

$$\frac{\Gamma_{opt}}{E[\Gamma_{JMQ}(R)]} \tag{3.29}$$

$$\geq \frac{MC + C + KC}{RC + MRC\frac{M+1}{M+1+K} + KC} \tag{3.30}$$

$$= \frac{M + MR}{MR - 1 + R + MR\frac{M+1}{M+MR}} \tag{3.31}$$

$$= \frac{M^2(R+1)^2}{(M+1)RM(R+1) - M(R+1) + MR(M+1)} \tag{3.32}$$

$$\to \frac{(R+1)^2}{R(R+1) + R} \tag{3.33}$$

$$= 1 + \frac{1}{R^2 + 2R} \tag{3.34}$$

as $M \to \infty$ and $K = MR - 1$. □

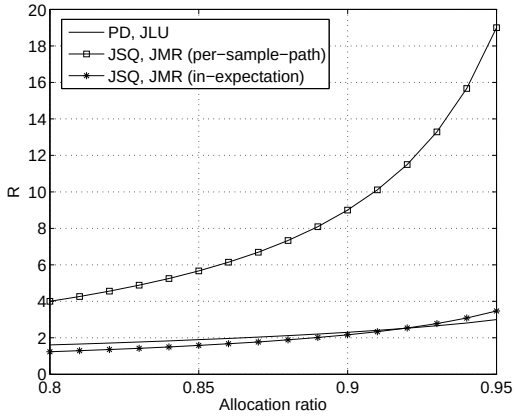

Figure 3.3: Capacity requirements of different policies.
Reprint with permission from [30].

3.6.3 Discussions

We have shown that our policies are $(R, \frac{e^R}{e^R-1})$-competitive-per-sample-path, while JSQ and JMR are no better than $(R, 1+\frac{1}{R})$-competitive-per-sample-path, and no better than $(R, 1+\frac{1}{R^2+2R})$-competitive-in-expectation. Suppose we are given a system where the offline policy can allocate all jobs. In order to guarantee that at least $1-\frac{1}{\theta}$ of the jobs are allocated, our policy only needs to increase the capacity by R times so that $\frac{e^R}{e^R-1} \leq 1/(1-\frac{1}{\theta})$. Therefore, choosing $R = \ln\theta$ is sufficient. In contrast, the two commonly used policies, JSQ and JMQ, require to increase the server capacity by at least $(\theta-1)$ times. Even when we consider that the arrival sequence is a random permutation of all jobs, and only require JSQ and JMQ to allocate $1-\frac{1}{\theta}$ of the jobs *on average*, they still need to increase the capacity by at least $\sqrt{\theta}-1$ times.

Fig. 3.3 plots the capacity requirement for different allocation ratios. From the figure we can observe that as the allocation ratio approaches 1, the capacity

requirement using JSQ or JMQ increases much faster than that using PD or JLU. For example, if we need to allocate at least 95% of the jobs, i.e., $\theta = 20$, our policies only require $R = 3$, while JSQ and JMQ both require $R \geq 19$. Even when the arrival sequence is a random permutation of all jobs, JSQ and JMQ still need $R \geq 3.5$. Further, one may notice that JSQ and JMQ seem to outperform our policies when the arrival sequence is a random permutation, and when the allocation ratio is low. However, the comparison is not fair. When the arrival sequence is a random permutation, our policies are likely to achieve better performance. In the next section, we will demonstrate this by simulations.

3.7 Revised Model With Buffer

In this section, we consider a more complicated model which uses a slotted time system. Similar to the previous one, the system has multiple non-identical servers and jobs arrive at the system sequentially. Jobs are of different types, and each job can only be served by a subset of the servers. Instead of serving the jobs or discard them instantly, we add buffer to each server. Job arrivals occur over time, and we assume that each job has a certain delay tolerance such that the job can be put into the buffer and wait for service. Their deadlines are arbitrary and bounded.

A server j has service capacity of C_j per time slot and buffer capacity of B_j. Each job takes unit service capacity to be processed and unit buffer capacity to wait for service. We use $a(i)$ to denote the arrival time of job i. When job i arrives, it reveals its server subset K_i and maximum waiting time $T(i)$. And waiting time is bounded by T, i.e. $T = max_i T(i)$. We use X_{ijt} to denote the service for job i. If $X_{ijt} = 1$, then job i is served by server j at time t. We use $k_{ijt} = 1$ to denote that job i can be served by server j and time t is within the delay tolerance, that is, if $k_{ijt} = 1$, then $j \in K_i$ and $a(i) \leq t \leq a(i) + T(i)$.

We aim to maximize the total number of jobs that can be served under the constraints of server capacity and buffer capacity. We formulate the problem as the following linear programming problem:

Revised Allocation:

$$Max \sum_{ijt} k_{ijt} X_{ijt} \tag{3.35}$$

$$s.t. \sum_{i} k_{ijt} X_{ijt} \leq C_j, \forall t, j \in \mathcal{J}, \tag{3.36}$$

$$\sum_{\substack{i, \\ t:\tau \in [a(i),t]}} k_{ijt} X_{ijt} \leq B_j, \forall \tau, j \in \mathcal{J}, \tag{3.37}$$

$$\sum_{jt} X_{ijt} \leq 1, \forall i \in \mathcal{I}, \tag{3.38}$$

$$X_{ijt} \geq 0, \forall i \in \mathcal{I}, j \in \mathcal{J}, t. \tag{3.39}$$

In the above problem, (3.35) is the total number of served jobs. (3.36) states that each server j can serve at most C_j jobs for each time slot. (3.37) states that at any time, the total number of jobs accepted and yet to be served by server j is at most B_j. (3.38) states that each job can be served at most once.

Similar to the previous case, we consider that the service provider can increase both capacities to allocate more jobs. When the capacities are increased by R times, the service capacity becomes RC_j and the buffer capacity becomes RB_j. We have our linear programming as follows:

Revised Allocation(R):

$$Max \sum_{ijt} k_{ijt} X_{ijt} \tag{3.40}$$

$$s.t. \sum_{i} k_{ijt} X_{ijt} \leq RC_j, \forall t, j \in \mathcal{J}, \tag{3.41}$$

$$\sum_{\substack{i, \\ t:\tau \in [a(i),t]}} k_{ijt} X_{ijt} \leq RB_j, \forall \tau, j \in \mathcal{J}, \tag{3.42}$$

$$\sum_{jt} X_{ijt} \leq 1, \forall i \in \mathcal{I}, \tag{3.43}$$

$$X_{ijt} \geq 0, \forall i \in \mathcal{I}, j \in \mathcal{J}, t. \tag{3.44}$$

To solve the **Revised Allocation(R)** problem, we first find the dual problem as:

Revised Dual:

$$Min \sum_{jt} C_j \alpha_{jt} + \sum_{j\tau} B_j \gamma_{\tau j} + \sum_{i} \beta_i, \tag{3.45}$$

$$s.t.\ k_{ijt} \alpha_{jt} + \sum_{\tau \in [a(i),t]} k_{ijt} \gamma_{\tau j} + \beta_i \geq k_{ijt}, \forall i \in \mathcal{I}, j \in \mathcal{J}, t \tag{3.46}$$

$$\alpha_{jt} \geq 0, \forall j \in \mathcal{J}, t, \tag{3.47}$$

$$\gamma_{j\tau} \geq 0, \forall j \in \mathcal{J}, \tau, \tag{3.48}$$

$$\beta_i \geq 0, \forall i \in \mathcal{I}, \tag{3.49}$$

Here each α_{jt} corresponds to a constraint in (3.41), each $\gamma_{j\tau}$ corresponds to a constraint in (3.42), and each β_i corresponds to a constraint in (3.43).

Now we introduce our online scheduling policy. There are two variables α_{jt} and $\gamma_{j\tau}$ that the policy maintains. They can be seen as the monitors for the usage of service capacity and buffer capacity. Initially, they are both set to be 0.

When a job i arrives, the policy checks the values of $k_{ijt}(1-\alpha_{jt}-\sum_{\tau\in[a(i),t]}\gamma_{\tau j})$ for all (j,t) pairs, and selects the pair (j^*,t^*) which maximizes the value of $k_{ijt}(1-\alpha_{jt}-\sum_{\tau\in[a(i),t]}\gamma_{\tau j})$. If $k_{ij^*t^*}(1-\alpha_{j^*t^*}-\sum_{\tau\in[a(i),t^*]}\gamma_{\tau j^*})>0$, job i is assigned to server j^*, waiting to be served at time t^*. Therefore $X_{ij^*t^*}=1$. The value of $\alpha_{j^*t^*}$ is updated to be $\alpha_{j^*t^*}(1+\frac{1}{C_{j^*}})+\frac{1}{(d_{j^*}-1)C_{j^*}}$, where we set $d_j=(1+1/C_j)^{RC_j}$, for all j. Each $\gamma_{\tau j^*}$, for $\tau\in[a(i),t^*]$, is updated to be $\gamma_{\tau j^*}\leftarrow\gamma_{\tau j^*}(1+\frac{1}{B_{j^*}})+\frac{1}{(f_{j^*}-1)B_{j^*}}$, where we set $f_j=(1+1/B_j)^{RB_j}$, for all j. On the other hand, if $k_{ij^*t^*}(1-\alpha_{j^*t^*}-\sum_{\tau\in[a(i),t^*]}\gamma_{\tau j^*})\le 0$, job i is discarded. The complete policy is described in Algorithm 6.

Algorithm 6 PD Algorithm for Revised Model

Initially, $\alpha_{jt}=0$, $\gamma_{j\tau}=0$, $X_{ijt}=0$.
$d_j\leftarrow(1+1/C_j)^{RC_j},\forall j$.
$f_j\leftarrow(1+1/B_j)^{RB_j},\forall j$.
for each arriving job i **do**
  $(j^*,t^*)\leftarrow\text{argmax}_{(j,t)}\,k_{ijt}(1-\alpha_{jt}-\sum_{\tau\in[a(i),t]}\gamma_{\tau j})$.
  if $k_{ij^*t^*}(1-\alpha_{j^*t^*}-\sum_{\tau\in[a(i),t^*]}\gamma_{\tau j^*})>0$ **then**
    $\beta_i\leftarrow 1-k_{ij^*t^*}(1-\alpha_{j^*t^*}-\sum_{\tau\in[a(i),t^*]}\gamma_{\tau j^*})$
    $\alpha_{j^*t^*}\leftarrow\alpha_{j^*t^*}(1+\frac{1}{C_{j^*}})+\frac{1}{(d_{j^*}-1)C_{j^*}}$.
    $\gamma_{\tau j^*}\leftarrow\gamma_{\tau j^*}(1+\frac{1}{B_{j^*}})+\frac{1}{(f_{j^*}-1)B_{j^*}},\forall\tau\in[a(i),t^*]$
    $X_{ij^*t^*}\leftarrow 1$.
    Job i is assigned to server j^*, to be served at t^*.
  else
    Discard job i.
  end if
end for

We first need to show that the vector $\{X_{ijt}\}$ produced by this policy satisfies all constraints of **Revised Allocation(R)**, so that the policy never assigns a job to a server that is already fully utilized.

Lemma 9. *Let $\alpha_{jt}[n]$ be the value of α_{jt} after n jobs are scheduled to be served by j at t. Let $\gamma_{\tau j}[n]$ be the value of $\gamma_{\tau j}$ when n jobs are waiting in server j at τ and to be served at a later time t. Then,*

$$\alpha_{jt}[n] = (\frac{1}{d_j - 1})(d_j^{n/RC_j} - 1). \tag{3.50}$$

$$\gamma_{\tau j}[n] = (\frac{1}{f_j - 1})(f_j^{n/RB_j} - 1). \tag{3.51}$$

Proof. Here we prove (3.50) and (3.51) by induction.

Initially, when $n = 0$, $\alpha_{jt}[0] = 0 = (\frac{1}{d_j - 1})(d_j^0 - 1)$ and (3.50) holds.

Suppose (3.50) holds when $n = k$. When the $(k+1)$-th job is scheduled to be served by j at t, we have

$$\begin{aligned}
\alpha_{jt}[k+1] =& \alpha_{jt}[k](1 + \frac{1}{C_j}) + \frac{1}{(d_j - 1)C_j} \\
=& \frac{1}{(d_j - 1)}(d_j^{k/RC_j} - 1)(1 + \frac{1}{C_j}) + \frac{1}{(d_j - 1)C_j} \\
=& \frac{1}{(d_j - 1)}[d_j^{(k+1)/RC_j} - 1],
\end{aligned}$$

and (3.50) still holds for $n = k + 1$. By induction, (3.50) holds for all n.

Similarly, when $n = 0$, $\gamma_{\tau j}[0] = 0 = (\frac{1}{f_j - 1})(f_j^0 - 1)$ and (3.51) holds.

Suppose (3.51) holds when $n = k$, i.e., there are k jobs waiting for service in server j. When the $(k+1)$-th job is scheduled to be served by j at time t, where $t > \tau$, we have

$$\begin{aligned}\gamma_{\tau j}[k+1] =& \gamma_{\tau j}[k](1+\frac{1}{B_j}) + \frac{1}{(f_j-1)B_j} \\ =& \frac{1}{(f_j-1)}(f_j^{k/RB_j}-1)(1+\frac{1}{B_j}) + \frac{1}{(f_j-1)B_j} \\ =& \frac{1}{(f_j-1)}[f_j^{(k+1)/RB_j}-1],\end{aligned}$$

and (3.51) still holds for $n = k+1$. By induction, (3.51) holds for all n. □

With Lemma 9, $\alpha_{jt} = 1$ when RC_j jobs have been scheduled to be served by j at t, and $\gamma_{\tau j} = 1$ when RB_j jobs are waiting in the buffer of server j at τ. In Algorithm 6, jobs are only allocates to servers with $k_{ijt}(1-\alpha_{jt}-\sum_{\tau\in[a(i),t]}\gamma_{\tau j}) > 0$, which guarantees $\alpha_{jt} \leq 1$ and $\gamma_{\tau j} \leq 1$. Thus our policy does not violate any constraints in **Revised Allocation(R)**.

Next, we study the competitive ratio of Algorithm 6.

Theorem 18. *Algorithm 6 is* $(R, \frac{e^R+T}{e^R-1})$*-competitive-per-sample-path and is* $(R, \frac{e^R+T}{e^R-1})$*-competitive-in-expectation.*

Proof. We prove Theorem 18 by three steps:

First, we show that solutions $\{\alpha_{jt}\}$, $\{\gamma_{\tau j}\}$ and $\{\beta_i\}$ satisfy all constraints in **Revised Dual**. Initially, α_{jt} and $\gamma_{\tau j}$ are set to be 0. By step 8 in Algorithm 6, α_{jt} and $\gamma_{\tau j}$ are non-decreasing. Hence (3.47) and (3.48)holds. Also, by Lemma 9, $\alpha_{jt} \leq 1$, $\gamma_{\tau j} \leq 1$, for all j. When a job i arrives, our policy selects $(j^*, t^*) \leftarrow \text{argmax}_{(j,t)}\, k_{ijt}(1-\alpha_{jt}-\sum_{\tau\in[a(i),t]}\gamma_{\tau j})$ according to step 5. When $(\sum_{\tau\in[a(i),t^*]}\gamma_{\tau j^*} + \alpha_{j^*t^*}+) < 1$, we have $k_{ijt}(\alpha_{jt}+\sum_{\tau\in[a(i),t]}\gamma_{\tau j}) \geq k_{ij^*t^*}(\alpha_{j^*t^*}+\sum_{\tau\in[a(i),t^*]}\gamma_{\tau j^*}) = 1-\beta_i$, then (3.46) and (3.49) hold. When $(\alpha_{j^*t^*}+\sum_{\tau\in[a(i),t^*]}\gamma_{\tau j^*}) = 1$, we have $(\alpha_{jt}+\sum_{\tau\in[a(i),t]}\gamma_{\tau j}) \geq 1$ and $\beta_i = 0$, (3.46) and (3.49) hold.

Next, we derive the ratio between (3.45) and (3.40). Both formulas are initially 0. When a job i arrives, we use $\Delta P(R)$ to denote the change of $\sum_{ijt} k_{ijt}X_{ijt}$, and ΔD to denote the change of $\sum_{jt} C_j\alpha_{jt} + + \sum_{j\tau} B_j\gamma_{\tau j} + \sum_i \beta_i$. If job i is discarded, both formulas remain unchanged, therefore $\Delta P(R) = \Delta D = 0$. If job i is scheduled to be serverd by j at t, then we have $X_{ijt} = 1$ and $\Delta P(R) = 1$. Also we have

$$\begin{aligned}
&\frac{\Delta D}{\Delta P(R)} = \Delta D \\
=&C_j(\frac{\alpha_{jt}}{C_j} + \frac{1}{(d_j-1)C_j}) + \sum_{\tau\in[a(i),t]} B_j(\frac{\gamma_{\tau j}}{B_j} + \frac{1}{(f_j-1)B_j}) \\
&+ 1 - \alpha_{jt} - \sum_{\tau\in[a(i),t]} \gamma_{\tau j} \\
=&1 + \frac{1}{d_j-1} + \sum_{\tau\in[a(i),t]} \frac{1}{f_j-1} \\
\leq&1 + \frac{1}{d_j-1} + \frac{T}{f_j-1}
\end{aligned}$$

When we imposes a lower bound on C_j and B_j by requiring $C_j \geq C_{min}$ and $B_j \geq B_{min}$, for all j, and let $C_{min} \to \infty$, $B_{min} \to \infty$, we have $d_j \to (e^R - 1)$ and $f_j \to (e^R - 1)$, and $\frac{\Delta D}{\Delta P(R)} \to \frac{e^R+T}{e^R-1}$, whenever a job i is allocated to some server. Therefore, we have, under Algorithm 6,

$$\frac{\sum_{jt} C_j\alpha_{jt} + + \sum_{j\tau} B_j\gamma_{\tau j} + \sum_i \beta_i}{\sum_{ijt} k_{ijt}X_{ijt}} \leq \frac{e^R+T}{e^R-1}.$$

Finally, by weak duality theorem, we establish that Algorithm 6 is $(R, \frac{e^R+T}{e^R-1})$-competitive-per-sample-path. By Lemma 3, Algorithm 6 is also $(R, \frac{e^R+T}{e^R-1})$-competitive-in-expectation. □

In order to guarantee that at least $1 - \frac{1}{\theta}$ of the jobs are allocated, Algorithm 6 needs to increase the capacity by R times so that $\frac{e^R+T}{e^R-1} \leq 1/(1 - \frac{1}{\theta})$. Therefore

$R = \ln(\theta T + \theta - T)$. If buffer is not bottleneck, then the competitive ratio is $(R, \frac{e^R}{e^R - 1})$. To guarantee that at least $1 - \frac{1}{\theta}$ of the jobs are allocated, we need $R = \ln \theta$. Specially, when $R = 1$, this competitive ratio achieve the optimal ratio in [53].

3.8 Simulation

In this section, we evaluate the performance of the four policies, including PD, JLU, JSQ, and JMQ by simulations. We consider three different scenarios:

First, we compare our polices with JSQ. We construct a system with two types of servers $J1,\ J2$; and two types of jobs $I1,\ I2$. Type $I1$ job can be served by type $J2$ server, type $I2$ job can be served by both $J1$ and $J2$ server. The number of jobs and capacities of servers are shown in Table 3.1. The arrival sequence is a random permutation of all jobs. Simulation results are the average of 10 runs. Under different R, we computer the allocation ratio by the number of jobs allocated with online policy dividing that with optimal offline policy. The simulation result is shown in Fig. 3.4. We can observe that PD and JLU policies outperform JSQ.

Table 3.1: System setting for the first scenario.
Reprint with permission from [30].

Server Type	Number of Servers	Capacity	Job Type	Number of Jobs	K_i
$J1$	1	25000	$I1$	2500	$J2$
$J2$	50	50	$I2$	25000	$J1,\ J2$

Next, we compare our polices with JMQ. We construct a system with two types of servers: $J1,\ J2$; and two types of jobs: $I1,\ I2$. The setting for jobs and servers are shown in Table 3.2. The result is shown in Fig. 3.5. We can observe that PD

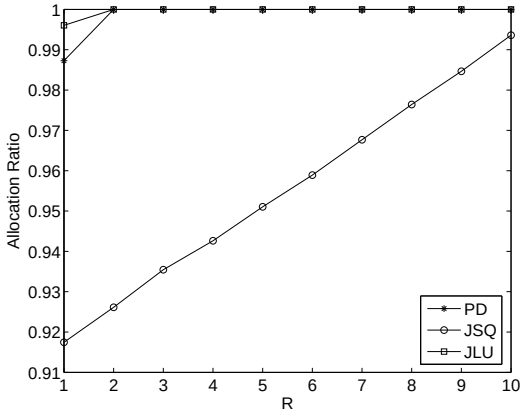

Figure 3.4: Simulation results for the first scenario.
Reprint with permission from [30].

Table 3.2: System setting for the second scenario.
Reprint with permission from [30].

Server Type	Number of Servers	Capacity	Job Type	Number of Jobs	K_i
$J1$	1	2550	$I1$	2550	$J1$
$J2$	500	50	$I2$	25000	$J1,\ J2$

and JLU policies outperform JMQ.

Last, we construct a system with four types of servers: $J1$, $J2$, $J3$, and $J4$; and four types of jobs: $I1,\ I2,\ I3,\ I4$. The detailed setting for servers and jobs are listed in Table 3.3, which is a combination of the two previous settings. Simulation results are shown in Fig. 3.6. We can observe that PD and JLU outperform both JSQ and JMQ.

From Fig. 3.4 to 3.6, we notice that our proposed PD and JLU policies almost have identical performance. Also, we notice that the allocation ratios of all four

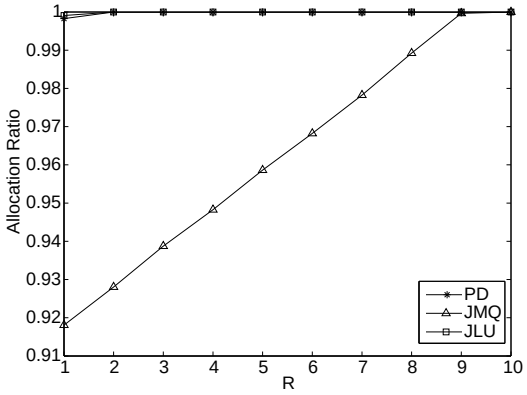

Figure 3.5: Simulation results for the second scenario.
Reprint with permission from [30].

Table 3.3: System setting for the third scenario.

Server Type	Number of Servers	Capacity	Job Type	Number of Jobs	K_i
$J1$	1	10000	$I1$	2500	$J2$
$J2$	50	50	$I2$	10000	$J1$,$J2$
$J3$	1	2550	$I3$	2550	$J3$
$J4$	200	50	$I4$	10000	$J3$,$J4$

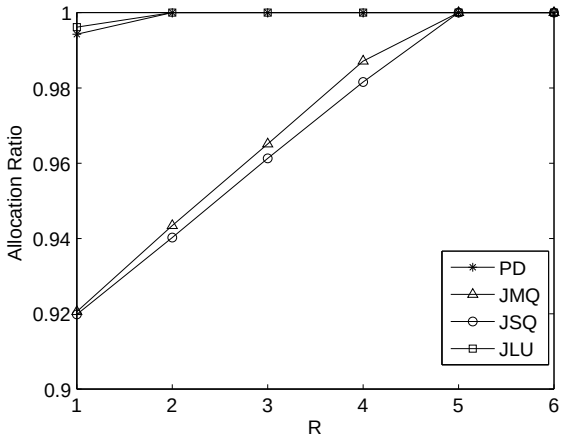

Figure 3.6: Simulation results for the third scenario.
Reprint with permission from [30].

policies converge to 1 as R increasing. However, JSQ and JMQ converges much slower than PD and JLU.

In Section 3.4 we prove that PD and JLU are $(R, \frac{e^R}{e^R-1})$-competitive-in-expectation. In Secion 3.6 we prove that both JSQ and JMQ are no better than $(R, 1 + \frac{1}{R^2+2R})$-competitive-in-expectation. Although it looks like that the competitive ratio in expectation of our policies is not as good as that of the other two policies when R is small, the simulation results show that our policies have better allocation ratio even with small R. Thus with random permutation of the jobs arrival sequence, the performance of our policies are better than of JSQ and JMR policies.

3.9 Conclusion

In this section, we have studied the job allocation problem with unknown job arriving pattern under hard allocation ratio requirement. Given the capacity of current data center which serves all jobs offline, we aim to find how much capacity we need to expand to meet the allocation ratio requirement for any unknown job arrival sequence.

We propose two online policies PD and JLU which are both $(R, \frac{e^R}{e^R-1})$-competitive-per-sample-path. We also prove that our policies can achieve any allocation ratio requirement with the least capacity. Next we study the performance of two widely used policies, JSQ and JMQ. We prove that both policies are no better than $(R, 1+\frac{1}{R})$-competitive-per-sample-path. Therefore, they need an order higher capacity to achieve the same allocation ratio requirement than our policies. We further prove that JSQ nd JMQ are no better than $(R, 1 + \frac{1}{R^2+2R})$-competitive-in-expectation by taking random permutation of the jobs in the arrival sequence. The simulation results show that our policies are still better than JSQ and JMQ. Thus our policies are much preferable than the two widely used JSQ and JMQ policies.

4. ONLINE ROUTING IN MULTI-HOP NETWORK WITH END-TO-END DEADLINE*

4.1 Problem Overview

Many emerging safety-critical applications, such as Internet of Things (IoT) and Cyber-Physical Systems (CPS), require communication protocols that support strict end-to-end delay and reliability guarantees for all packets. In a typical scenario, when sensors detect unusual events that can cause system instability, they send out this information to actuators or control centers. This information needs to be delivered within a strict deadline for actuators or control centers to resolve the unusual events. The system can suffer from a critical fault when a small portion of packets fail to be delivered on time.

Despite the huge literature on quality of service (QoS), there is little work that can provide end-to-end delay and reliability guarantees simultaneously, especially when packet arrivals are time-varying and unpredictable. The lack of progress is mainly caused by two fundamental challenges. On one hand, it is obvious that one cannot design the optimal network policies without obtaining complete knowledge of future packet arrivals and incurring high computation complexity. Therefore, practical solutions need to rely on online suboptimal policies. On the other hand, in a multi-hop network, the scheduling decision of one communication link will impact the decisions of subsequent links. The negative effects of suboptimal decisions by online policies therefore get accumulated along the path of multi-hop transmissions. In fact, a recent work by Mao, Koksal, and Shroff [54] has proved

*Reprinted with permission from "On the capacity requirement for arbitrary end-to-end deadline and reliability guarantees in multi-hop networks" by Han Deng, I-Hong Hou, 2017, ACM Sigmetrics. [55]

that the performance of any online policies deteriorates as the length of the longest path in the network increases. As a result, no online policy can provide meaningful performance guarantees when the size of the network is large.

In order to maintain desirable performance using online suboptimal policies, current practice is to add redundancy into the system. During system deployment, the capacities of communication links are chosen to be larger than necessary. Such redundancy alleviates the negative impacts of suboptimal decisions by online policies. Using this approach, a critical question is to determine the amount of redundancy needed to provide the desirable performance guarantees. This section aims to answer this question [55].

We first show that the problem of maximizing the number of timely packet deliveries can be formulated as a linear programming problem when one knows the complete knowledge of all future packet arrivals. In the setting of online policies, some of the parameters of this linear programming problem will only be revealed when the corresponding packets arrive. Therefore, online policies need to make routing and scheduling decisions for packets without knowing all parameters. On the other hand, we also observe that adding redundancy by increasing link capacities is equivalent to relaxing a subset of constraints in the linear programming problem. Based on these observations, we define a competitive ratio that, given the amount of redundancy, quantifies the relative performance of online policies in comparison to the optimal offline solution.

Using the primal-dual method, we propose an online policy that achieves good performance in terms of competitive ratio. This policy has several important features: First, when there is no redundancy added to the system, the performance of our online policy is asymptotically better than that of the recent work [54] when the size of the network increases. Second, we also show that only a small amount

of redundancy is needed to achieve strict performance guarantees. Specifically, in order to guarantee the timely delivery of at least $1 - \frac{1}{\theta}$ as many packets as the optimal solution in a network whose longest path has length L, our policy only needs to increase link capacities by $\ln L + \ln \theta$ times. Finally, we also show that our policy can be implemented with very low complexity.

Next, we establish a theoretical lower bound of competitive ratio for all online policies. We show that, in order to guarantee a certain degree of performance, the redundancy needed by our policy is only a small amount away from the theoretical limit. In particular, when both L and θ, as defined in the previous paragraph, go to infinity, the redundancy needed by our policy is at most twice as large as the theoretical limit.

We also study online policies when one cannot increase network capacity by adding redundancy. We propose another online policy and prove that it is order optimal with fixed link capacity. Specifically, we show that this online policy guarantees to deliver at least $\frac{1}{O(\log_L)}$ as many packets before their deadlines as the optimal offline solution, where L is the maximum route length. As the previous study [54] has proved no online policy can deliver more than $\frac{1}{O(\log_L)}$ packets without redundancy, our policy is order-optimal.

While neither of our online policies need any information about future packet arrivals to make routing and scheduling decisions, they are centralized algorithms that require tight coordination. For large networks without a centralized coordinator, we also propose a fully distributed protocol that is inspired by the design principles of our centralized online policies. This distributed protocol only requires each node to broadcast its local congestion information very infrequently, and therefore it only incurs a small amount of communication overhead. When a packet arrives at a source node, the source node determines a suggested route for

the packet using its received congestion information, and each link on the route makes scheduling decisions solely based on its local information.

All three of our policies are evaluated by simulations. We compare our policies with the widely used earliest deadline first policy (EDF) and recent policy studied in [54]. Simulation results show that all our policies perform better than the other two policies. This result is in particular surprising because our distributed protocol even achieves better performance than the online policy in [54], which is a centralized one.

4.2 Related Work

Online scheduling problem in real-time environment has been studied in many previous works. Studies show that earliest deadline first algorithm (EDF) [56, 57] and least laxity first algorithm (LLF) [57] achieve the same performance as the optimal offline algorithm when the system is under-loaded, that is, the optimal offline algorithm can serve all jobs in the system. In under-loaded system, all jobs enter the system can be served by EDF and we do not need to drop any job when it arrives at the system. However, in over-loaded system, even with optimal offline algorithm, there are still some jobs that cannot be served. EDF and LLF achieve the same performance as the optimal online policy when the system is over-loaded. Also [57] proved that no online algorithm can guarantee to serve more than $1/4$ of the jobs that can be served by optimal offline algorithm and provided an algorithm in a uniprocessor system which achieves $1/4$ service bound. [58, 59] consider admission control in online scheduling. In [58], when all jobs have equal length, the competitive ratio of deterministic algorithm is bounded by 2. [60] considers the similar model as [58]. It introduces a parameter k to indicate the willingness of a job to have a delay before being served. It shows that when

all jobs have equal length, the competitive ratio of deterministic algorithm is $(1 + 1/(\lfloor k \rfloor + 1))$-competitive instead.

In addition, online scheduling with multiple-server case has also been studied. [61] studies the scheduling of equal length jobs on two identical machines. [62–64] studies the case with parallel machines. The scheduler need to decide whether to accept or reject a packet and which machines is chosen to serve the job. [63] has proposed an algorithm with immediate decision which approaches $\frac{e}{e-1}$-competitive when the number of machines is greater or equal to 3. It also provide another lower bound that deterministic online algorithm with immediate decision is no better than 1.8-competitive when there are 2 machines. Later [64] has shown that online algorithm which makes immediate decision upon job releasing is bounded by $\frac{e}{e-1}$-competitive for multiple machine case.

There are also many works studying the scheduling problem in multihop network. An early study [65] focuses on the problem of packet scheduling with arbitrary end-to-end delay, fix route, and known packet injection rate. It propose a distributed algorithm which achieve a certain delay bound. [66] studies the scheduling problem on a tree network. Packets arrive at an arbitrary node and they need to be transmitted to root node before the deadlines. Any packet that cannot arrive root node within deadline is considered lost. Thus this is also a fix route problem. The goal is to minimize the total lost packets. Shortest time to extinction (STE) algorithm is proposed and it is shown to achieve the performance of optimal offline policy. Also there are many works studying the end-to-end delay in multiohop network. Rodoplu *et al.* [67] have studied the problem of dynamic estimating end-to-end delay over multi-hop mobile wireless networks. Sanada Komuro and Sekiya [68] have used Markov-chain model to study the string-topology multi-hop network and analyse the end-to-end throughput and delay. Jiao *et al.* [69] have

studied the problem of estimating the end-to-end delay distribution for general traffic arrival model and Nakagami-m channel model by analyzing packet delay at each hop. Li *et al.* [70]] have proposed using expected end-to-end delay for selecting path in wireless mesh networks. The expected end-to-end delay takes both queuing delay and delay caused by unsuccessful wireless transmissions. However, their work only aims at minimizing the average end-to-end delays, and cannot provide guarantees on per-packet delays.

Li and Eryilmaz [71] has studied the end-to-end deadline constrained traffic scheduling in multihop network. They develop algorithms to meet the deadline and throughput requirement in a wired network. However, they only consider the fix route model and they do not provide any performance guarantee. Wang *et al.* [72] have studied the problem of routing and scheduling on multi-hop wireless sensor network in order to optimize the system with the constraint of end-to-end delay and proposed a sub-optimal algorithm. Hou [73] proposed a throughput optimal policy for up-link tree networks with end-to-end delay constraints and delivery ratio requierement. The packets deadlines are the end of the frames in which they are generated. Singh and Kumar [74] have proposed a scheduling policy which maximize the throughput for multi-hop wireless networks. However, the paper uses a fix-route model and does not consider end-to-end delay. Mao, Koksal and Shroff [54] also considers a fix route problem. The network has arbitrary packet arrival and packet weight. The paper aims to maximize the total cumulative weight of packets that reach destination before their deadline. The paper has proved that the competitive ratio of any online policy is no better than $O(\log L)$, where L is the length of the maximum route. It has also proposed an admission control and packet scheduling policy and shown that it is $O(L \log L)$-competitive. Liu and Yang [75] have studied the multi-hop routing problem with hard end-to-

end delay and the throughput region. They also assume that packets are required to be delivered to destination within one frame and the performance is evaluated by simulation. Our work will focus on online routing and scheduling on multi-hop network with end-to-end delay constraint and aim to guarantee both packet deadline and network delivery ratio.

4.3 System Model

We consider a network with multihop transmissions. The network is represented by a directed graph where each node represents a router and an edge from one node to another represents a link between the corresponding routers. Packets arrive at their respective source nodes following some unknown sequence. We use $\mathcal{M}$ to denote the set of all packets and $\mathcal{L}$ the set of all links. When a packet $m \in \mathcal{M}$ arrives at its source node, it specifies its destination and a deadline. The packet requests to be delivered to its destination before its specified deadline. Packets that are not delivered on time do not have any value, and can be dropped from the network. We aim to deliver as many packets on time as possible.

We assume that time is slotted and numbered by $t = \{1, 2, 3, \dots\}$. Different links in the network may have different link capacities, and we denote by C_l the number of packets that link l can transmit in a time slot. At the beginning of each time slot, each node decides which packets to transmit over its links, subject to capacity constraints of the links. Packets transmitted toward a node in time slot t will be received by that node at the end of the time slot, so that the node can transmit these packets to subsequent nodes starting from time slot $t + 1$.

Delivering a packet to its destination before its deadline require determining two things: the route used to forward the packet from its source to its destination, and the times at which the packet is transmitted along its route. We define a *valid*

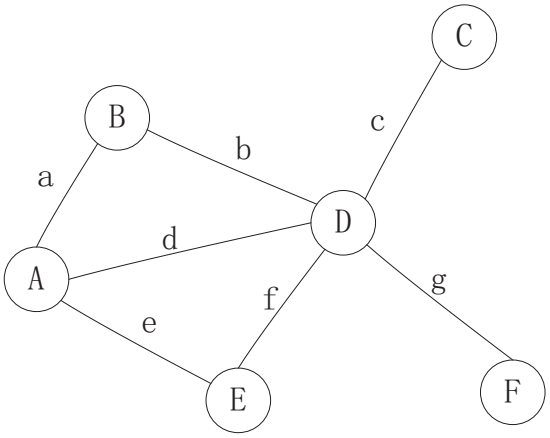

Figure 4.1: Network topology.

schedule for each packet m as the collection of links of a route, as well as the times of transmissions for each of these links, so that packet m can be delivered to its destination on time. For example, consider the network shown in Fig. 4.1. Suppose a packet arrives at node A at time slot 1, and needs to be delivered to node F before the end of time slot 3. One valid schedule for this packet is to transmit it over link d in time slot 1, and then over link g in time slot 2. We use $\{(d,1),(g,2)\}$ to represent this valid schedule. Other valid schedules include $\{(d,1),(g,3)\}, \{(e,1),(f,2),(g,3)\}$, etc. On the other hand, $\{(d,1),(g,4)\}$ is not a valid schedule because the packet is delivered to its destination after its deadline at time slot 4. The schedule $\{(d,3),(g,2)\}$ is not valid because it would require node D to transmit the packet over link g at time slot 2 before it receives the packet at time slot 3. For each packet m, we let $V(m)$ denote the set of valid schedules for m. The problem of deciding how to deliver packets on time then becomes one of choosing valid schedules for packets.

We use X_{mk} to denote the schedule selection for packet m. If $X_{mk} = 1$, packet m is transmitted using valid schedule k, and $X_{mk} = 0$, otherwise. Given the information of all packets, the problem of maximizing the total number of successful deliveries can be formulated as the following linear programming problem:

Schedule:

$$Max \sum_{m,k:k\in V(m)} X_{mk} \tag{4.1}$$

$$s.t. \sum_{k:k\in V(m)} X_{mk} \leq 1, \forall m \in \mathcal{M}, \tag{4.2}$$

$$\sum_{m,k:(l,t)\in k} X_{mk} \leq C_l, \forall l \in \mathcal{L}, t \in \{1, 2, \dots\}, \tag{4.3}$$

$$X_{mk} \geq 0, \forall m \in \mathcal{M}, k \in V(m). \tag{4.4}$$

Since $X_{mk} = 1$ if packet m is transmitted using valid schedule k, Eq. (4.1) is the total number of packets that are delivered on time. Eq. (4.2) states that at most one valid schedule can be chosen for each packet. Eq. (4.3) states that each link can transmit at most C_l packets in any time slot. In practice, X_{mk} can only be either 0 or 1, but our problem formulation allows X_{mk} to be any real number in $[0, 1]$. Thus, the optimal solution to **Schedule** describes an upper bound on the total number of successful deliveries.

If information of all packets is available when the system starts, the optimal solution to **Schedule** can be found by standard linear programming methods. In practice, however, packets arrive sequentially, and we need to rely on online policies that determines the values of X_{mk} for each arriving packet m without knowing future packet arrivals. Without the knowledge of future arrivals, it is obvious that online policies cannot always achieve the optimal solution to **Schedule**. In fact, a

recent work [54] has shown that, when the longest path between a source node and a destination node is L, no online policy can guarantee to deliver more than $\frac{1}{\log_2 L}$ as many packets as the optimal solution. To put this number in perspective, consider a medium-sized network with $L = 8$. Even when the optimal solution can deliver all packets on time, the bound in the recent work states that no online policy can guarantee to deliver more than $\frac{1}{\log_2 8} = \frac{1}{3}$ of all packets. Such performance of online policies is unacceptable for virtually any applications.

In order to achieve good performance for online policies in the presence of unknown future arrivals, we consider the scenario where service providers can increase link capacities by, for example, upgrading network infrastructures. When the link capacities are increased by R times, link l can transmit RC_l packets in each time slot. With the increase in capacities, our problem can be rewritten as follows:

Schedule(R):

$$Max \sum_{m,k:k\in V(m)} X_{mk} \tag{4.5}$$

$$s.t. \sum_{k:k\in V(m)} X_{mk} \leq 1, \forall m \in \mathcal{M}, \tag{4.6}$$

$$\sum_{m,k:(l,t)\in k} X_{mk} \leq RC_l, \forall l \in \mathcal{L}, t \in \{1, 2, \dots\}, \tag{4.7}$$

$$X_{mk} \geq 0, \forall m \in \mathcal{M}, k \in V(m). \tag{4.8}$$

To evaluate the performance of online policies, we define a competitive ratio that incorporates the increase in capacities:

Definition 4. *Given a sequence of packet arrivals, let Γ_{opt} be the optimal value of $\sum_{mk:k\in V(m)} X_{mk}$ in **Schedule**, and $\Gamma_\eta(R)$ be the number of packets that are delivered under an online policy η when the link capacities are increased by R times. The online*

policy η is said to be (R, ρ)-competitive *if* $\Gamma_{opt}/\Gamma_{\eta}(R) \leq \rho$*, for any sequence of packet arrivals.*

4.4 An Online Algorithm and Its Competitive Ratio

4.4.1 Algorithm Description

In this section, we propose an online policy based on primal-dual method and analyze the competitive ratio. We first note that the dual problem of **Schedule** is:

Dual:

$$Min \sum_{m} \alpha_m + \sum_{l,t} C_l \beta_{lt}, \tag{4.9}$$

$$s.t.\ \alpha_m + \sum_{l,t:(l,t)\in k} \beta_{lt} \geq 1, \forall m \in \mathcal{M}, k \in V(m) \tag{4.10}$$

$$\alpha_m \geq 0, \forall m, \tag{4.11}$$

$$\beta_{lt} \geq 0, \forall l, t, \tag{4.12}$$

where α_m is the Lagrange multiplier corresponding to constraint (4.2), and β_{lt} is the Lagrange multiplier corresponding to constraint (4.3).

By the Weak Duality Theorem, we have the following lemma:

Lemma 10. *Given any vectors of* $\{\alpha_m\}$ *and* $\{\beta_{lt}\}$ *that satisfy the constraints (4.10)–(4.12), we have* $\sum_m \alpha_m + \sum_{(l,t)} C_l \beta_{lt} \geq \Gamma_{opt}$.

We now introduce our online algorithm. Our algorithm constructs three variables $\{X_{mk}\}, \{\alpha_m\}, \{\beta_{lt}\}$ simultaneously while ensuring they satisfy all constraints in **Schedule(**R**)** and **Dual**. Initially, it sets $\beta_{lt} \equiv 0$. When a packet m arrives, the algorithm finds the valid schedule k^* that has the largest value of $(1 - \sum_{l,t:(l,t)\in k} \beta_{lt})$ among all $k \in V(m)$. If $1 - \sum_{l,t:(l,t)\in k^*} \beta_{lt} \leq 0$, then the algorithm drops packet

m and sets $\alpha_m = 0$ and $X_{mk} = 0$, for all $k \in V(m)$. On the other hand, if $1 - \sum_{l,t:(l,t)\in k^*} \beta_{lt} > 0$, packet m is transmitted using the valid schedule k^*. Our algorithm sets $X_{mk^*} = 1$, $\alpha_m = 1 - \sum_{l,t:(l,t)\in k^*} \beta_{lt}$, and updates β_{lt} as $\beta_{lt} = \beta_{lt}(1 + \frac{1}{C_l}) + \frac{1}{(d_l-1)C_l}$ for all $(l,t) \in k^*$, where d_l is chosen to be $(1+1/C_l)^{RC_l}$. The complete policy is shown in Algorithm 7.

Algorithm 7 Online Algorithm with Variable R

1: Initially, $\alpha_m \leftarrow 0$, $\beta_{lt} \leftarrow 0$, $X_{mk} \leftarrow 0$.
2: $d_l \leftarrow (1 + 1/C_l)^{RC_l}, \forall l$.
3: **for** each arriving packet m **do**
4: $k^* \leftarrow \text{argmax}_k \, (1 - \sum_{(l,t)\in k} \beta_{lt})$
5: **if** $(1 - \sum_{(l,t)\in k^*} \beta_{lt}) > 0$ **then**
6: $\alpha_m \leftarrow (1 - \sum_{(l,t)\in k^*} \beta_{lt})$
7: $\beta_{lt} \leftarrow \beta_{lt}(1 + \frac{1}{C_l}) + \frac{1}{(d_l - 1)C_l}, \ (l,t) \in k^*$
8: $X_{mk^*} \leftarrow 1$.
9: Transmit packet m using valid schedule k^*.
10: **else**
11: Drop packet m.
12: **end if**
13: **end for**

4.4.2 Complexity of the Algorithm

In step 4, the algorithm finds the valid schedule k^* that maximizes $(1-\sum_{l,t:(l,t)\in k} \beta_{lt})$. We now show that this step can be completed in polynomial time by dynamic programming. Before presenting the algorithm, some new notations are given as follows. We say that packet m joins the network at the beginning of time slot a_m, and specifies its deadline as f_m. Its source node and destination node are s_m and d_m, respectively. Therefore, a valid schedule for m is one that can deliver a packet from node s_m to node d_m between time slots a_m and f_m.

Let $\Theta(n,\tau)$ be the smallest value of $\sum_{l,t:(l,t)\in k}\beta_{lt}$ among all schedules that can deliver a packet from node s_m to node n between time slots a_m and τ. $\Theta(n,\tau)=\infty$ if there is no schedule that delivers a pacekt from s_m to d_m between time slots a_m and τ_m. Step 4 of Alg. 7 is then equivalent to finding the valid schedule that achieves $\sum_{l,t:(l,t)\in k}\beta_{lt}=1-\Theta(d_m,f_m)$. Since packet m arrives at the beginning of time slot a_m, or, equivalently, at the end of time slot a_m-1, we set $\Theta(s_m,a_m-1)=0$ and $\Theta(n,a_m-1)=\infty$ for $\forall n\neq s_m$.

There are only two different ways to deliver a packet to node n by the end of time slot τ: The first is to deliver the packet to n by time slot $\tau-1$, in which case $\sum_{l,t:(l,t)\in k}\beta_{lt}=\Theta(n,\tau-1)$. The second is to deliver the packet to one of n's neighbors, say, node q, by time slot $\tau-1$, and then forward the packet along the link l_{qn} from q to n at time slot τ. In this case, $\sum_{l,t:(l,t)\in k}\beta_{lt}=\Theta(q,\tau-1)+\beta_{l_{qn}\tau}$. Therefore, we have

$$\Theta(n,\tau)=min\begin{cases}\Theta(n,\tau-1),\\ \Theta(q,\tau-1)+\beta_{l_{qn}\tau},\ q \text{ is a neighbor of } n.\end{cases}$$

Based on the above recursive equation, we design an algorithm for computing $\Theta(n,\tau)$. The detailed algorithm is shown in Algorithm 8, where we also use $Sch(n,\tau)$ to denote the schedule that achieves $\Theta(n,\tau)$.

In Alg. 8, the inequality $\Theta(q,\tau-1)+\beta_{l_{qn}\tau}<\Theta(n,\tau)$ is only evaluated once for any link and time slot. Let E be the number of links in the system. Suppose the number of links is larger than the number of nodes, and $f_m-a_m+1\leq T$, for all m, then the complexity of Alg. 8 is $O(ET)$.

Algorithm 8 Dynamic Programming

for each arriving packet m **do**
  $\Theta(s_m, a_m - 1) \leftarrow 0$
  $\Theta(n, a_m - 1) \leftarrow \infty, \forall n \neq s_m$
  $Sch(n, a_m - 1) \leftarrow \phi, \forall n$
  for $\tau = a_m$ **to** f_m **do**
    for node n **do**
      $\Theta(n, \tau) \leftarrow \Theta(n, \tau - 1)$
      $Sch(n, \tau) \leftarrow Sch(n, \tau - 1)$
      for node n's neighbor q **do**
        if $\Theta(q, \tau - 1) + \beta_{l_{qn}\tau} < \Theta(n, \tau)$ **then**
          $\Theta(n, \tau) \leftarrow \Theta(q, \tau - 1) + \beta_{l_{qn}\tau}\}$
          $Sch(n, \tau) \leftarrow Sch(q, \tau - 1) \cup \{(l_{qn}, \tau)\}$
        end if
      end for
    end for
  end for
end for

4.4.3 Competitive Ratio Analysis

Before analyzing the performance of Algorithm 7, we first establish a basic property of the values of β_{lt}.

Lemma 11. *Let $\beta_{lt}[n]$ be the value of β_{lt} after n packets are scheduled to use link l at time t. Then,*

$$\beta_{lt}[n] = (\frac{1}{d_l - 1})(d_l^{n/RC_j} - 1). \tag{4.13}$$

Proof. First, note that the value of β_{lt} is only changed when Algorithm 7 uses link l at time t to transmit a packet. Therefore, the value of β_{lt} only depends on the number of packets that are scheduled to use link l at time t.

We then prove (4.13) by induction. Initially, when $n = 0$, $\beta_{lt}[0] = 0 = (\frac{1}{d_l - 1})(d_l^0 - 1)$ and (4.13) holds.

Suppose (4.13) holds for the first n packets. When the $(n+1)$-th packet is scheduled for link l at time t, we have

$$\begin{aligned}\beta_{lt}[n+1] =& \beta_{lt}[n](1+\frac{1}{C_l}) + \frac{1}{(d_l-1)C_l} \\ =& \frac{1}{(d_l-1)}(d_l^{n/RC_l}-1)(1+\frac{1}{C_l}) + \frac{1}{(d_l-1)C_l} \\ =& \frac{1}{d_l-1}[d_l^{n/RC_l}(1+\frac{1}{C_l})-1]\end{aligned}$$

We select $d_l = (1+\frac{1}{C_l})^{RC_l}$, and therefore

$$\beta_{lt}[n+1] = \frac{1}{(d_l-1)}[d_l^{(n+1)/RC_l}-1],$$

and (4.13) still holds for $n+1$. Thus, by induction, (4.13) holds for all n. □

We now establish the competitive ratio of Algorithm 7.

Theorem 19. *Let $C_{min} := \min C_l$, $d_{min} := (1+1/C_{min})^{RC_{min}}$, and L be the longest path between a source node and a destination node, that is, all valid schedules have $|k| \le L$, for all $m \in \mathcal{M}, k \in V(m)$. Algorithm 7 produces solutions that satisfy all constraints in **Schedule(R)** and **Dual**. Moreover, Algorithm 7 is $(R, 1+\frac{L}{d_{min}-1})$-competitive, which converges to $(R, 1+\frac{L}{e^R-1})$-competitive, as $C_{min} \to \infty$.*

Proof. First, we show that the dual solutions $\{\alpha_m\}$ and $\{\beta_{lt}\}$ satisfy constraints (4.10) to (4.12). Initially, we have $\beta_{lt} = 0$. By Lemma 11, $\beta_{lt} \ge 0$ holds. Since step 6 is only used when $(1 - \sum_{(l,t)\in k^*} \beta_{lt}) > 0$, $\alpha_m \ge 0$ holds. From step 4 and 6, we know that $\alpha_m + \sum_{(l,t)\in k} \beta_{lt} \ge (1 - \sum_{(l,t)\in k} \beta_{lt}) + \sum_{(l,t)\in k} \beta_{lt} = 1$. Thus (4.10) to (4.12) hold.

Next, we show $\{X_{mk}\}$ satisfies constraints (4.6) to (4.8). By step 4, the algorithm picks at most one schedule k^* for packet m, constraint (4.6) holds. With Lemma 11, $\beta_{lt} = 1$ when RC_l packets use link l at time t. Since a valid schedule including (l,t) will be chosen for packet m only when $(1 - \sum_{(l,t)\in k^*} \beta_{lt}) > 0$, all (l,t) in the chosen valid schedule must have $\beta_{lt} < 1$, and therefore the number of packets transmitted over link l at time t must be less than RC_l. Thus, at any time t, there are at most RC_l packets using link l. Constraint (4.7) holds. By initialization and step (8), constraint (4.8) holds.

We derive the ratio between $\sum_m \alpha_m + \sum_{(l,t)} C_l\beta_{lt}$ and $\sum_{mk} X_{mk}$. Initially, both are equal to 0. We consider the increasing amount for both when a new packet m arrives at the network. We use $\Delta P(R)$ to denote the change of $\sum_{mk} X_{mk}$, and ΔD to denote the change of $\sum_m \alpha_m + \sum_{(l,t)} C_l\beta_{lt}$.

If packet m is dropped, both $\Delta P(R)$ and ΔD are 0. If packet m is accepted and transmitted using valid schedule k^*, we have $X_{mk^*} = 1$. Thus, $\Delta P(R) = 1$. On the other hand, ΔD is increased as:

$$\begin{aligned}
\Delta D =& \alpha_m + \sum_{(l,t)\in k^*} C_l \Delta\beta_{lt} \\
=& (1 - \sum_{(l,t)\in k^*} \beta_{lt}) + \sum_{(l,t)\in k^*} (\beta_{lt} + \frac{1}{(d_l - 1)C_l}) \\
=& 1 + \sum_{(l,t)\in k^*} \frac{1}{(d_l - 1)} \leq 1 + \frac{L}{d_{min} - 1}
\end{aligned}$$

Therefore, for each packet arrival, the ratio between ΔD and $\Delta P(R)$is no larger than $1 + \frac{L}{d_{min}-1}$ if $\Delta D > 0$. When the algorithm terminates, we have

$\frac{\sum_m \alpha_m + \sum_{(l,t)} C_l \beta_{lt}}{\sum_{mk} X_{mk}} \leq 1 + \frac{L}{d_{min}-1}$. By Lemma 10, $\frac{\Gamma_{opt}}{\sum_{mk} X_{mk}} \leq 1 + \frac{L}{d_{min}-1}$, and the competitive ratio of Algorithm 7 is $(R, 1 + \frac{L}{d_{min}-1})$. When $C_{min} \to \infty$, $d_{min} = (1 + \frac{1}{C_{min}})^{RC_{min}} \to e^R$, and the competitive ratio of Algorithm 7 converges to $(R, 1 + \frac{L}{e^R-1})$. □

There are several important implications of Theorem 19. First, without increasing capacity, that is, when $R = 1$, the competitive ratio of our policy is $(1, O(L))$. In comparison, the online algorithm proposed in the recent work [54] focuses on the special case of $R = 1$ and has a competitive ratio of $(1, O(L \log L))$. Therefore, our algorithm is asymptotically better than the online algorithm in [54]. Second, this theorem allows us to quantify the amount of capacity needed to a certain performance guarantee. Suppose the optimal solution to **Schedule** indeed delivers all packets. In order to guarantee that $1 - \frac{1}{\theta}$ of the packets are transmitted to their destinations before their deadlines, Theorem 19 states that we only need to increase all link capacities by R_θ times so that $1 + \frac{L}{e^{R_\theta}-1} \leq 1/(1 - \frac{1}{\theta}) = 1 + \frac{1}{\theta-1}$. Therefore, we have $R_\theta = \ln(L(\theta - 1) + 1) \leq \ln L + \ln \theta$. For example, if we are required to deliver 99% of the packets and the longest path consists of 10 hops, then we need to increase capacity by 6.9 times.

4.5 A Theoretical Lower Bound for Competitive Ratio

In Section 4.4, we showed that our policy is $(R, 1 + \frac{L}{e^R-1})$-competitive. In this section, we will establish a lower bound for the competitive ratio of online policies.

Theorem 20. *Any online algorithm cannot be better than* $(R, 1+\frac{L-2e^R}{(L+1)e^R-L})$*-competitive.*

Proof. We design a network as shown in Fig 4.2. We start to construct the network from an up-link tree, which is shown as the white nodes in Fig 4.2. Root is marked as node D and it is the destination of all packets. There are N levels of non-root

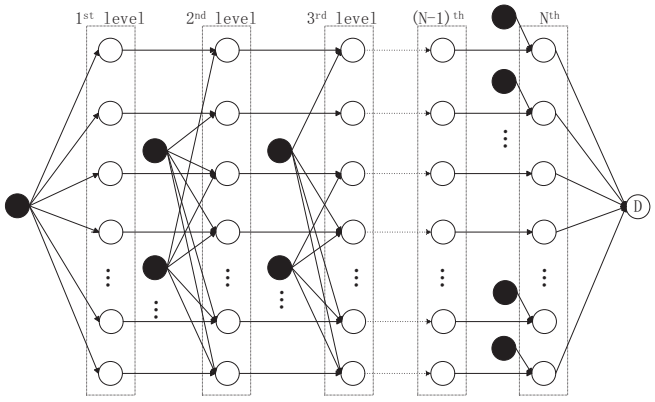

Figure 4.2: Network topology for lower bound analysis

nodes with N nodes in each level. Each node is connected to one node in the next level. Nodes do not share parent except the N-th level nodes share the same root node. At the j-th level, where $1 \leq j \leq N$, there are $\binom{N}{N+1-j}$ extra nodes, which is shown as the black nodes in Fig 4.2, with each node connecting to an unique set of $N+1-j$ nodes in this level. For example, there is one black node connected to all white nodes in level 1, and there are N black nodes connected to white nodes in level 2, where each of these black nodes is connected all but one white nodes in level 2. Likewise, there are $\binom{N}{N-2}$ black nodes connected to white nodes in level 3, with each black node connected to $N-2$ white nodes in level 3, and no two black nodes are connected to the same subset of white nodes.

Next, we describe packet arrivals. Packets only arrive at black nodes. Of all black nodes connected to the same level of white nodes, only one black node has packet arrival. Let $\mathcal{W}_j$ be the set of white nodes in j-th level which connects to the black node with packet arrivals. The black nodes with packet arrivals are chosen such that all nodes in $\mathcal{W}_{j+1}$ are connected to those in $\mathcal{W}_j$. Fig 4.3 is a simplified network of Fig 4.2, where we omit the black nodes with no packet arrival and

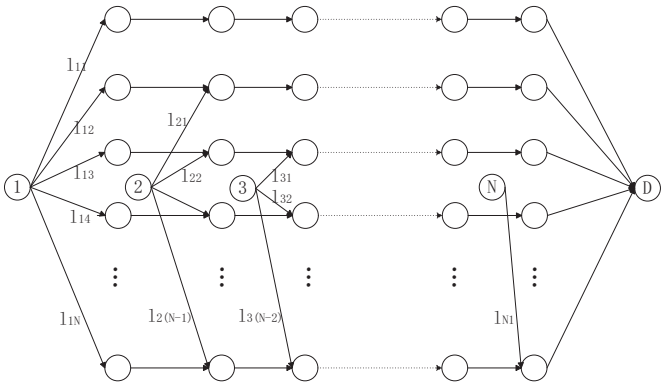

Figure 4.3: Simplified network topology for lower bound analysis

marked each black node with a number from 1 to N.

Packets arrive at nodes $1, 2, ..., N$. Their destination is node D. Each link in the network has capacity C. At the beginning of time slot 1, there are C packets arriving at node 1. Node 1 is connected to N links: l_{11}, l_{12}, $\cdots$, l_{1N}. At the beginning of time slot 2, there are C packets arriving at node 2. Node 2 is connected to $N-1$ links: l_{21}, l_{22}, $\cdots$, $l_{2(N-1)}$. Similarly for nodes 3, 4, $\cdots$. At the beginning of time N, there are C packets arriving at node N. The deadline of all packets is $N+1$. Node N is connected only to link l_{N1}.

When one knows which black nodes have packet arrivals, the offline optimal algorithm is to transmit the first C packets through link l_{11} and the following links, the second C packets through link l_{21} and the following links, $\ldots$, and the N-th C packets through link l_{N1} and the following link. The total number of delivered packets is NC.

Next we consider the online algorithm when all links' capacity is increased by R times. Since online policies do not know which black nodes will have packet arrivals, the optimal online policy is to distribute packets evenly among all connected

links. That is, at time 1, each of links l_{1i}, $i = 1, 2, \cdots, N$, transmit C/N packets. At time 2, each of link l_{2i}, $i = 1, 2, \cdots, (N-1)$, transmits $C/(N-1)$ packets. At time K, link l_{Ki}, $i = 1, 2, \cdots, (N-K+1)$, transmits $C/(N-K+1)$ packets. For simplicity, we call the routes from node 1 to node D through l_{1i} route r_i. If all packets arrive at node K are accepted, routes r_i, $i = K, K+1, \cdots, N$ have the same load on each link. When any link on a single route reaches its capacity, the route cannot be used for future arrival packets. Suppose the route gets over-loaded at time $K+1$, that is, packets arrive at node K are accepted and packets arrive at node $K+1$ are not fully accepted. The maximum load of a single link on route r_N is at most $\frac{C}{N} + \frac{C}{N-1} + \cdots + \frac{C}{N-K+1}$ and at least $\frac{C}{N} + \frac{C}{N-1} + \cdots + \frac{C}{N-K}$. We then have:

$$C(\frac{1}{N} + \frac{1}{N-1} + \frac{1}{N-2} + \ldots + \frac{1}{N-K+1}) \leq RC,$$

and

$$C(\frac{1}{N} + \frac{1}{N-1} + \frac{1}{N-2} + \ldots + \frac{1}{N-K}) \geq RC.$$

Since

$$\int_{N-K+1}^{N+1} \frac{1}{x} dx < (\frac{1}{N} + \frac{1}{N-1} + \frac{1}{N-2} + \ldots + \frac{1}{N-K+1}),$$

and

$$\int_{N-K-1}^{N} \frac{1}{x} dx > (\frac{1}{N} + \frac{1}{N-1} + \frac{1}{N-2} + \ldots + \frac{1}{N-K}).$$

We have:

$$\log(N+1) - \log(N-K+1) = \log \frac{N+1}{N-K+1} < R,$$

and

$$\log(N) - \log(N - K - 1) = \log \frac{N}{N - K - 1} > R.$$

Then we can derive the value of K as: $N - \frac{N}{e^R} - 1 \leq K \leq N + 1 - \frac{N+1}{e^R}$. The total number of accepted packets is in the range $((N - \frac{N}{e^R} - 1)C, (N + 2 - \frac{N+1}{e^R})C)$.

Thus the competitive ratio of an online policy is at best $(R, \frac{N}{N+2-\frac{N+1}{e^R}})$. In Fig. 4.2, the longest path in the network is between the leftmost black node and the sink, which has length $L = N + 1$. The competitive ratio can then be rewritten as $(R, 1 + \frac{L-2e^R}{(L+1)e^R-L})$. □

Let us once again consider the scenario where online policies need to guarantee to deliver at least $1 - \frac{1}{\theta}$ as many packets as the optimal solution. Theorem 20 states that any online policy needs to increase its link capacities by at least R_θ times so that $1 + \frac{L-2e^{R_\theta}}{(L+1)e^{R_\theta}-L} \leq 1 + \frac{1}{\theta-1}$. Solving this equation, and we have R_θ needs to be at least $\ln L + \ln \theta - \ln(L + 2\theta - 1)$. In comparison, our policy only needs to increase link capacities by $(\ln L + \ln \theta)$ times to ensure the delivery of $1 - \frac{1}{\theta}$ as many packets as the optimal solution. Therefore, the capacity requirement of our policy is at most $\ln(L + 2\theta - 1)$ away from the lower bound. Suppose we fix the ratio between L and θ, and let them both go to infinity, then we have $(\ln L + \ln \theta)/(\ln L + \ln \theta - \ln(L + 2\theta - 1)) \to 2$. Therefore, when both L and θ are large, our policy at most requires twice as much capacity as the theoretical lower bound.

4.6 An Order-Optimal Online Policy with Fixed $R = 1$

We have shown that Alg. 7 is $(R, 1 + \frac{L}{d_{min}-1})$-competitive. Without increasing link capacity, i.e, $R = 1$, the algorithm is $(1, 1 + \frac{L}{e-1})$-competitive, as $C_{min} \to \infty$. While the competitive ratio of Alg. 7 is an order better than that of the online

policy in the previous work [54], it still fails to achieve the theoretical bound of $(1, O(\log L))$-competitive. In this section, we propose another online algorithm and prove that it achieves the theoretical bound when $R = 1$.

4.6.1 Algorithm Description

Similar to the design of Alg. 7, we aim to design an algorithm that constructs $\{X_{mk}\}, \{\alpha_m\}, \{\beta_{lt}\}$ while ensuring they satisfy all constraints in **Schedule** and **Dual**. The algorithm is described in Alg. 9. One can see that Alg. 9 is very similar to Alg. 7, and their only difference lie in the update rules for β_{lt}. Specifically, let $\beta_{lt}[n]$ be the value of β_{lt} when link l serves a total number of n packets at time t. Then Alg. 9 chooses the value of $\beta_{lt}[n]$ as:

$$\beta_{lt}[n] = \begin{cases} \dfrac{1}{L(e^{\frac{1}{\ln L+1}} - 1)}(e^{\frac{n}{C_l}} - 1), \text{if } n \leq \frac{C_l}{\ln L+1}; \\ e^{(\frac{n}{C_l}-1)(\ln L+1)}, \text{if } n \geq \frac{C_l}{\ln L+1}. \end{cases} \tag{4.14}$$

To illustrate the difference in β_{lt}, we plot the values of $\beta_{lt}[n]$ for a link with $C_l = 1000$ under the two policies in Fig. 4.4, where we consider the two cases $L = 8$ and $L = 64$ for Alg. 9. As can be shown in the figure, when n is small, Alg. 9 increases the value of β_{lt} much slower than Alg. 7 does. Moreover, Alg. 9 increases β_{lt} slower when L is larger. Recall that both Alg. 7 and Alg. 9 only schedule a packet when $\max_k (1 - \sum_{(l,t)\in k} \beta_{lt}) > 0$, or, equivalently, $\min_k \sum_{(l,t)\in k} \beta_{lt} < 1$. By increasing β_{lt} slower when n is small, Alg. 9 ensures that more packets with long routes can be accepted, especially when the network is lightly loaded.

4.6.2 Competitive Ratio Analysis

We now prove that Alg. 9 achieves the theoretical bound in [54] by being $(1, O(\log L))$-competitive.

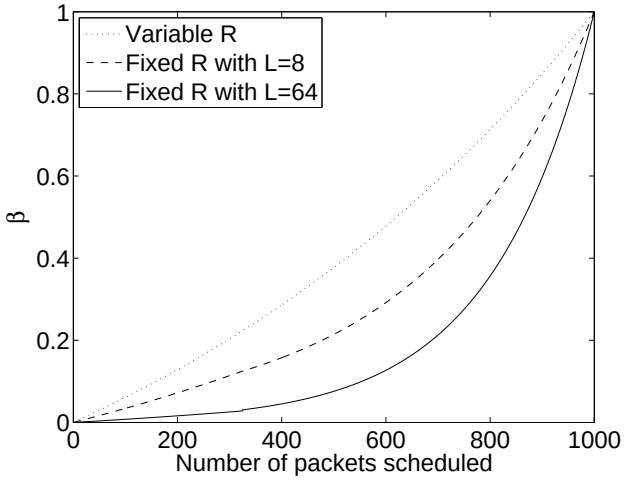

Figure 4.4: Values of β_{lt} under different policies.

Algorithm 9 Online Algorithm with Fixed $R = 1$

Initially, $\alpha_m \leftarrow 0$, $\beta_{lt} \leftarrow 0$, $X_{mk} \leftarrow 0$.
for each arriving packet m **do**
  $k^* \leftarrow \text{argmax}_k \left(1 - \sum_{(l,t)\in k} \beta_{lt}\right)$
  if $\left(1 - \sum_{(l,t)\in k^*} \beta_{lt}\right) > 0$ **then**
    $\alpha_m \leftarrow \left(1 - \sum_{(l,t)\in k^*} \beta_{lt}\right)$
    for each $(l,t) \in k^*$ **do**
      if total number of packets n at time t on link l: $n \leq \frac{C_l}{\ln L+1}$ **then**
        $\beta_{lt} \leftarrow \frac{1}{L(e^{\frac{1}{\ln L+1}} - 1)}(e^{\frac{n}{C_l}} - 1),$
      else
        $\beta_{lt} \leftarrow e^{(\frac{n}{C_l}-1)(\ln L+1)}$
      end if
    end for
    $X_{mk^*} \leftarrow 1.$
    Transmit packet m using valid schedule k^*.
  else
    Drop packet m.
  end if
end for

Lemma 12. *Let $C_{min} := \min C_l$. In Algorithm 9, each time a new packet is scheduled, the ratio between the change of* ***Schedule*** *and* ***Dual*** *is bounded by $2(\ln L + 1) + \frac{B}{C_{min}}$, where the value of B is independent of C_{min}.*

Proof. If a new packet is admitted to the network, the increasing amount of **Dual** is

$$\begin{aligned}\Delta D =& \alpha_m + \sum_{(l,t)\in k^*} C_l \Delta\beta_{lt} \\ =& 1 + \sum_{(l,t)\in k^*} (C_l \Delta\beta_{lt} - \beta_{lt})\end{aligned}$$

We define $\beta(x)$ as

$$\beta(x) = \begin{cases} \dfrac{1}{L(e^{\frac{1}{\ln L+1}} - 1)}(e^x - 1), \text{if } x \le \frac{1}{\ln L+1}; \\ e^{(x-1)(\ln L+1)}, \text{if } x \ge \frac{1}{\ln L+1}. \end{cases} \tag{4.15}$$

Note that $\beta_{lt}[n] = \beta(\frac{n}{C_l})$. By using Taylor Sequence, we then have

$$\begin{aligned}\Delta\beta_{lt}[n] :=& \beta_{lt}[n+1] - \beta_{lt}[n] = \beta(\frac{n+1}{C_l}) - \beta(\frac{n}{C_l}) \\ \le& \frac{1}{C_l}\beta'(\frac{n}{C_l}) + \epsilon\frac{1}{C_l^2}\beta''(\frac{n}{C_l}),\end{aligned}$$

for some bounded constant $\epsilon < \infty$, where β' and β'' are the first and second derivative of β, respectively. We note that the function $\beta(x)$ is continuous for all x, and infinitely differentiable for all x except at the point $x_0 := \frac{1}{\ln L+1}$. At the point x_0, we define $\beta'(x_0) = \lim_{x\to x_0^+} \beta'(x)$ and $\epsilon\beta''(x_0) = \lim_{x\to x_0^+} \epsilon\beta''(x)$. This ensures that the

above inequality still holds.

By (4.14) we know that $n \leq \frac{C_l}{\ln L+1}$ if and only if $\beta_{lt}[n] \leq \frac{1}{L}$.

If $x = \frac{n}{C_l} \leq \frac{1}{\ln L+1}$, then $\beta'(x) = \beta''(x) = \frac{e^x}{\frac{1}{L(e^{\frac{1}{\ln L+1}}-1)}}$. We have:

$$\begin{aligned}
&C_l\Delta\beta_{lt}[n] - \beta_{lt}[n] \\
\leq &\frac{C_l(\frac{1}{C_l}e^{\frac{n}{C_l}}) + \epsilon(\frac{1}{C_l})^2e^{\frac{n}{C_l}}) - (e^{\frac{n}{C_l}} - 1)}{L(e^{\frac{1}{\ln L+1}} - 1)} \\
\leq &\frac{1 + \epsilon\frac{1}{C_l}e^{\frac{n}{C_l}}}{L(1 + \frac{1}{\ln L+1} - 1)} \\
\leq &\frac{\ln L + 1}{L}(1 + \epsilon\frac{1}{C_l}e)
\end{aligned}$$

Let $B_1 = \epsilon e\frac{\ln L+1}{L}$, then

$$C_l\Delta\beta_{lt}[n] - \beta_{lt}[n] \leq \frac{\ln L + 1}{L} + B_1\frac{1}{C_{min}}, \tag{4.16}$$

when $\frac{n}{C_l} \leq \frac{1}{\ln L+1}$.

On the other hand, If $x = \frac{n}{C_l} \geq \frac{1}{\ln L+1}$, then $\beta'(x) = (\ln L + 1)\beta(x)$ and $\beta''(x) = (\ln L + 1)^2\beta(x)$. We have:

$$\begin{aligned}
&C_l\Delta\beta_{lt}[n] - \beta_{lt}[n] \\
\leq &C_l[\frac{\ln L + 1}{C_l}\beta_{lt}[n] + \epsilon(\frac{\ln L + 1}{C_l})^2\beta_{lt}[n]] - \beta_{lt}[n] \\
\leq &\ln L \cdot \beta_{lt}[n] + \frac{1}{C_l}\epsilon(\ln L + 1)^2\beta_{lt}[n]
\end{aligned}$$

Let $B_2 = \epsilon(\ln L + 1)^2$, then

$$C_l \Delta\beta_{lt}[n] - \beta_{lt}[n] \leq (\ln L + B_2 \frac{1}{C_{min}})\beta_{lt}[n], \tag{4.17}$$

when $\frac{n}{C_l} \geq \frac{1}{\ln L+1}$.

If packet m is transmitted using valid schedule k^*, we have $X_{mk^*} = 1$. Thus, $\Delta P = 1$. On the other hand, ΔD is increased as:

$$\begin{aligned}
\Delta D =& 1 + \sum_{(l,t):(l,t)\in k^*} C_l \Delta\beta_{lt} - \beta_{lt} \\
\leq& 1 + \sum_{(l,t):(l,t)\in k^*, \beta_{lt} \leq \frac{1}{L}} C_l \Delta\beta_{lt} - \beta_{lt} \\
&+ \sum_{(l,t):(l,t)\in k^*, \beta_{lt} \geq \frac{1}{L}} C_l \Delta\beta_{lt} - \beta_{lt}
\end{aligned}$$

From (4.16) and (4.17) we have:

$$\begin{aligned}
\Delta D \leq& 1 + \sum_{(l,t):(l,t)\in k^*, \beta_{lt} \leq \frac{1}{L}} (\frac{\ln L + 1}{L} + B_1 \frac{1}{C_{min}}) \\
&+ \sum_{(l,t):(l,t)\in k^*, \beta_{lt} \geq \frac{1}{L}} ((\ln L + B_2 \frac{1}{C_{min}})\beta_{lt})
\end{aligned}$$

From Algorithm 9 step 4 we know that $\sum \beta_{lt} \leq 1$, thus we have

$$\begin{aligned}
\Delta D \leq& 1 + (\ln L + 1 + B_1 \frac{L}{C_{min}}) + (\ln L + B_2 \frac{1}{C_{min}}) \\
=& 2 + 2\ln L + \frac{B_1 + B_2}{C_{min}},
\end{aligned}$$

and the proof is complete. □

Theorem 21. *Algorithm 9 produces solutions that satisfy all constraints in* ***Schedule*** *and* ***Dual****. Moreover, it is* $(1, 2(1 + \ln L))$*-competitive, as* $C_{min} \to \infty$.

Proof. First, we show that the dual solutions $\{\alpha_m\}$ and $\{\beta_{lt}\}$ satisfy constraints (4.10) to (4.12). Initially, we have $\beta_{lt} = 0$. By (4.14), $\beta_{lt} \geq 0$ holds. Since step 5 is only used when $(1 - \sum_{(l,t)\in k^*} \beta_{lt}) > 0$, $\alpha_m \geq 0$ holds. From step 3 and 5, we know that $\alpha_m + \sum_{(l,t)\in k} \beta_{lt} \geq (1 - \sum_{(l,t)\in k} \beta_{lt}) + \sum_{(l,t)\in k} \beta_{lt} = 1$. Thus (4.10) to (4.12) hold.

Next, we show $\{X_{mk}\}$ satisfies constraints (4.2) to (4.4). By step 3, the algorithm picks at most one schedule k^* for packet m, constraint (4.2) holds. With (4.14), when the number of packets on link l at t is C_l, we have $\beta_{lt} = 1$. Also, since a packet is scheduled if $(1 - \sum_{(l,t)\in k^*} \beta_{lt}) > 0$, we have $\beta_{lt} < 1$ for all $(l, t) \in k^*$. Therefore, the number of packets transmitted on link l at any time t is at most C_l. Constraint (4.3) holds. By initialization and step (13), constraint (4.4) holds.

When a new packet m arrives, it will either be dropped or scheduled. If it is dropped, both ΔP and ΔD are 0. If it is scheduled, both (4.9) and (4.1) increase. With Lemma 12,the ratio between ΔP and ΔD is bounded by $2(1 + \ln L) + \frac{B}{C_{min}}$. Therefore the competitive ratio of Algorithm 9 is $(1, 2(1 + \ln L) + \frac{B}{C_{min}}) \to (1, 2(1 + \ln L))$, as $C_{min} \to \infty$.

□

Thus, comparing with the result in [54], Algorithm 9 achieves the optimal competitive ratio when $R = 1$.

4.7 A Fully Distributed Protocol for Implementation

The two algorithms that we have proposed so far are both centralized algorithms. Specifically, when a packet arrives at a node, the node needs to have complete knowledge of all β_{lt} of all links to find a valid schedule. Such information

is usually infeasible to obtain. In this section, we propose a distributed protocol based on the design of Algorithm 7.

In our distributed protocol, the task of transmitting a packet to its destination is decomposed into two parts: First, when a packet arrives at a node, the node determines a suggested schedule based on statistics of past system history. This suggested schedule consists of the route for forwarding the packet, as well as a local deadline for each link. After determining the suggested schedule, the node simply forwards it to the first link of the route. On the other hand, when a link receives a packet along with a suggested schedule, the link tries to forward the packet to the next link in the suggested schedule before its local deadline. The link drops the packet when it cannot forward the packet on time.

To facilitate this protocol, each link keeps track of its own β_{lt}, which reflects the number of packets that are scheduled to be transmitted over link l at time t. The value of β_{lt} changes over time, as link l schedules more and more packets to be transmitted at time t. Therefore, we define $\beta_{lt,\hat{t}}$ as the value of β_{lt} when the current time is $\hat{t}$. Each link then measures $\gamma_{l,\tau}$ as the average of $\beta_{lt,t-\tau}$. In other words, when the current time is t_0, the expected value of β_{lt} is $\gamma_{l,t-t_0}$. Link l broadcasts its $\gamma_{l,\tau}$ periodically so that all nodes can estimate the values of β_{lt}.

We now describe how a node determines a suggested schedule upon the arrival of a packet. Suppose a packet arrives at time t_0. Following the Alg. 8, the node would like to find a valid schedule that maximizes $(1 - \sum_{l,t:(l,t)\in k} \beta_{lt})$. In practice, the node does not know the exact value of β_{lt}. However, it knows that the expected value of β_{lt} is $\gamma_{l,t-t_0}$. In our protocol, the node assumes that $\beta_{lt} = \gamma_{l,t-t_0}$, and then finds a valid schedule k^* that maximizes $(1 - \sum_{(l,t)\in k} \gamma_{l,t-t_0})$. Similar to Alg. 8, the node drops the packet if $(1 - \sum_{(l,t)\in k^*} \gamma_{l,t-t_0}) \leq 0$. If $(1 - \sum_{(l,t)\in k^*} \gamma_{l,t-t_0}) > 0$, then the node puts information of k^* into the header of the packet, and forwards the

packet to the first link in k^*.

Algorithm 10 Distributed Implementation: Schedule Suggestion for Each Node

for each arriving packet m **do**
  $t_0 \leftarrow$ current time
  $k^* \leftarrow \text{argmax}_k \left(1 - \sum_{(l,t)\in k} \gamma_{l,t-t_0}\right)$
  if $\left(1 - \sum_{(l,t)\in k} \gamma_{l,t-t_0}\right) > 0$ **then**
    Put information of the suggested schedule k^* in the header of packet m.
    Forward the packet to the first link in k^*.
  else
    Drop packet m.
  end if
end for

Since the actual value of β_{lt} can be different from $\gamma_{l,t-t_0}$, there is no guarantee that a packet can be delivered on time using the valid schedule k^* even if $\left(1 - \sum_{(l,t)\in k^*} \gamma_{l,t-t_0}\right) > 0$. Therefore, when a node determines a valid schedule k^* for a packet, the valid schedule k^* is treated only as a suggestion for links in k^*. Specifically, if k^* contains an entry (l^*, t^*), then the link l^* interprets k^* as a requirement that l^* needs to forward the packet to the next link before t^*, or drops the packet. When l^* obtains the packet, it still has the freedom to choose when to forward the packet, as long as the packet is forwarded before time t^*.

Next, we discuss how each link determines the actual time to transmit each packet. Obviously, each link l^* knows its own β_{l^*t}. From the design of Alg. 8, we can see that Alg. 8 prefers to transmit packets when β_{lt} is small. Our proposed policy is based on this principle. When a link l^* receives a packet, it finds the entry (l^*, t^*) from the valid schedule k^* specified in the header of the packet. Link l^* then finds a time t_{tx} between the current time and t^* that has the smallest β_{l^*t}, and transmits the packet at time t_{tx}. Alg. 11 describes the details of the policy for

packet transmission.

Algorithm 11 Distributed Implementation: Packet Transmission for Each Link

for each packet m **do**
  Upon m's arrival at a link l^*, the link reads schedule information k^* from the header of the packet. Let t^* be the local deadline such that $(l^*, t^*) \in k^*$.
  $t^*_{\text{tx}} \leftarrow \text{argmin}_{t_{\text{tx}}: t_{\text{tx}} \leq t^*} \beta_{l^*, t_{\text{tx}}}$
  if $\beta_{l^*, t^*_{\text{tx}}} < 1$ **then**
    $\beta_{l^* t^*_{\text{tx}}} \leftarrow \beta_{l^*, t^*_{\text{tx}}}(1 + \frac{1}{C_{l^*}}) + \frac{1}{(d_{l^*} - 1)C_{l^*}}$
    Transmit packet m on link l^* at time t^*_{tx}.
  else
    Drop packet m.
  end if
end for

4.8 Simulation

In this section, we evaluate the performance of our policies by simulation. We compare our algorithms with EDF policy and the policy, which we call Mao-Koksal-Shroff (MKS) online algorithm, proposed in [54]. Both EDF policy and MKS online algorithm focus on packet scheduling, and are applicable only when the route of the packet is given. For these two policies, we assume that each packet is routed through the shortest path.

We first consider a small network as shown in Fig 4.5. The network has 9 nodes from node 1 to node 9. There are directed arrows showing the directed links between nodes. All links have the same capacity $C = 1$. We assume that there are 1000 packets arriving at the system. For each packet, the source node is chosen uniformly at random between node 1 to node 6, and the destination is chosen uniformly at random between node 7 to node 9. The inter-arrival time between

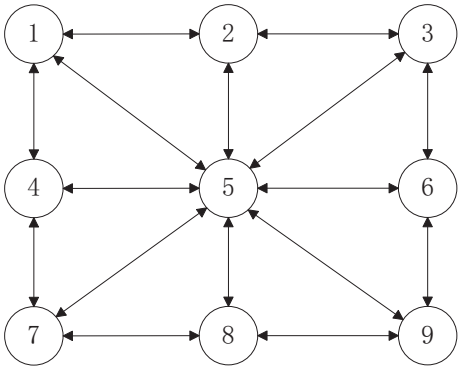

Figure 4.5: Network topology for a small network

packets are chosen to be 0 with probability 0.7 and 1 with probability 0.3. The deadline of each packet equals its arrival time plus a slack time. The slack time is chosen uniformly from integers between 2 and 6.

Simulation results for different values of R are shown in Fig. 4.6. From the result, we can see that all our three policies outperform two other current policies. From the figure, we can see that all our policies are able to deliver all packets when R is 2. On the other hand, EDF is able to deliver all packets when $R = 3$, and MKS can deliver all packets only when R is as large as 6. We also note that both EDF and MKS are centralized policies. The fact that our distributed algorithm performs better than these two centralized policies further highlights the superiority of our algorithms.

Next, we consider that different links can have different capacities. Since MKS requires all links to have the same capacity, we only compare our policies against EDF. The network topology is also shown in Fig 4.5. We assume that, when $R = 1$, the link capacity are integers uniformly chosen from 5 to 10. There are 10000 i.i.d

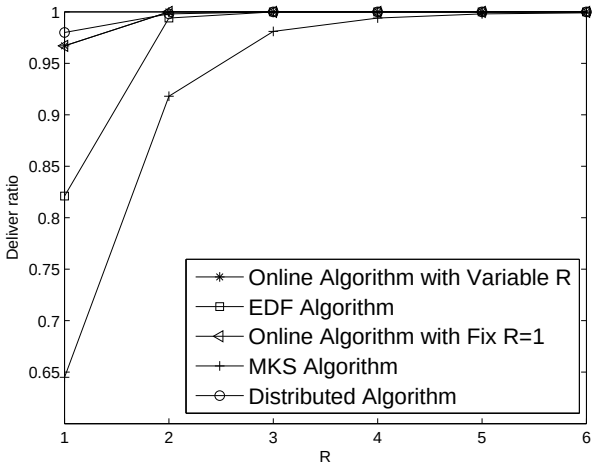

Figure 4.6: Deliver ratio comparison when all links have the same capacity.

packets to be delivered. At the beginning of each time slot, there are a certain number of packets arriving the system. The source node and destination node are both chosen from node 1 to node 9 with equal probability and destination node is not allowed to be the same with source node. The number of packets is randomly chosen between 100 and 500. Each packet has a slack time between arrival and deadline, which is uniformly chosen from $[2, 6]$. The result is shown in Fig 4.7. Once again, we see that our policies, including the distributed algorithm, perform much better than EDF in most cases.

Finally, we simulate and compare the policies in a 7×7 grid network as shown in Fig. 4.8. The link capacity are integers uniformly chosen from 1 to 10. At the beginning of each time slot, packets arrive at node 1, 4, and 7. Each node receive 49 packets and the packet are evenly delivered to node 43 to node 49 with relative deadlines from 15 to 21, respectively. The system is tested for a running time of 300 time slots. In this large grid network, we also compare our policies with EDF.

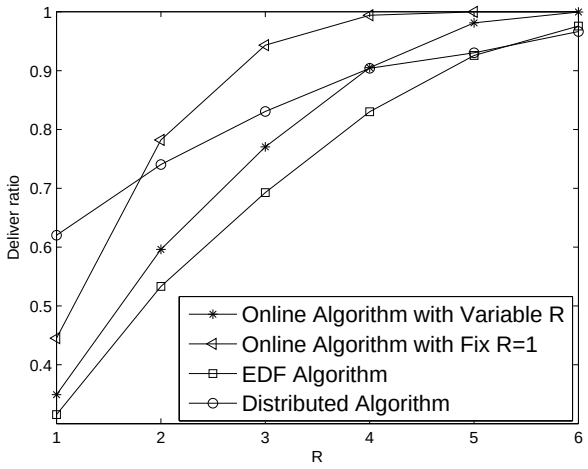

Figure 4.7: Deliver ratio comparison when links have different capacities.

The result is shown in Fig 4.9.

From the result, we can see that as the origin and destination node increase, the routing gets more complicated, and number of packets increases, the congestion on the network increases. In this setting, both our policies outperform EDF policy. Also, the distributed policy provide a satisfied delivery ratio. With smaller link capacity, the distributed algorithm even outperform all three other policies.

4.9 Conclusion

In this section, we study the multi-hop network scheduling problem with end-to-end deadline and hard transmission rate requirement. Given the capacity of each link in the network, we aim to find out how much capacity we need to increase to guarantee the required ratio of packets can be successfully transmitted to its destination before its deadline without knowing the packet arrival sequences in advance.

We have proposed an online algorithm which works for both fix route and non-

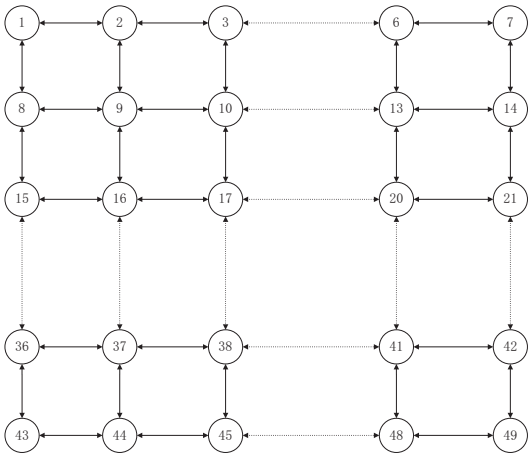

Figure 4.8: Network topology for a 7 by 7 network

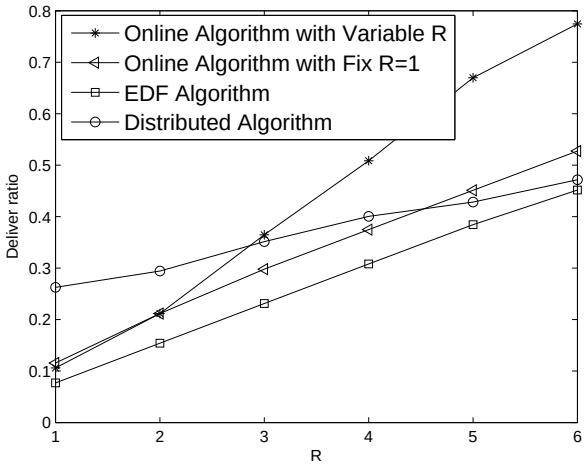

Figure 4.9: Deliver ratio in grid network

fix route network. The algorithm is proved to be $(R, 1 + \frac{L}{e^R - 1})$-competitive, where L is the length of the longest path. We have also showed that the complexity of our algorithm is $O(ET)$, where E is the total number of links and T is the largest slack time. Next, we have showed that any online algorithm cannot be better than $(R, 1 + \frac{L - 2e^R}{(L+1)e^R - L})$-competitive. When both L and required deliver rate are large, our policy requires at most twice as much capacity as the lower bound. In addition, We have proposed an online algorithm for fixed capacity network. When the capacity cannot be increased, our algorithm is proved to be $(1, O(\log L))$-competitive, which is also an order-optimal policy. For practical implementation of our centralized algorithm, we have proposed a heristic for distributed algorithm so that each node can make decisions without requiring real-time information from all other nodes. In addition to the theoretical results, we compare our policies with two other online policies, including the widely-used EDF policy and a recent proposed policy, by simulation. The results show that the performance of our policies are better than the other two policies. Also the result shows that the distributed algorithm still provide a good delivery ratio.

5. CONCLUSION

In the dissertation, we aim to explore the trade-off between system redundancy and system performance in online scheduling. We define a new competitive ratio to quantify the system performance with the increased system redundancy, instead of simply considering the performance ratio of an online policy and the offline optimal policy.

We have studied three network applications and explored the new defined competitive ratio for each application.

The first application is delayed mobile offloading, in which case WiFi is used to reduce mobile traffic when users have unpredictable movement pattern. We propose two online algorithms and study their competitive ratio. We show that they achieve optimal performance and have better performance than other three commonly used algorithms. To achieve same offload ratio, our policies only need half as much capacity as the three commonly used policies.

Second, we study online job allocation problem. Jobs arrive the system in sequence and each job reveals its service constraint upon arrival. The constraint include server subset and deadline. We design two online policies to maximize the total jobs served by the system. With the two policies, we study how much server capacity is needed to guarantee a certain allocation ratio of all arriving jobs. Then we study the performance of optimal online policy and prove that the proposed policies are optimal. We also compare our proposed policies with two other commonly used policies. We consider both worst-case arrival sequence and random job arrival sequences. We show that our policies need much less capacity to achieve the same performance of the other two policies in the two cases.

Last, we study the online routing in multi-hop end-to-end network. Packets arrive the network in sequence and reveal destination node and deadline upon arrival. We propose online policies which work for both fix route and non-fix route network. We show that our policies have better performance than the policy of a recent study. Also, based on the centralized policy, we propose an heuristic distributed algorithm for practical implementation. In the distributed algorithm, each node receive the information of the whole network periodically and make routing decision based on the received information. One future problem is to consider if there is an optimal policy which jointly considers the delivery rate and capacity increasing.

REFERENCES

[1] Cisco, "Cisco visual networking index: Global mobile data traffic forecast update, 2013âĂŞ2018," 2014.

[2] J. G. Andrews, H. Claussen, M. Dohler, S. Rangan, and M. C. Reed, "Femtocells: Past, present, and future," *Selected Areas in Communications, IEEE Journal on*, vol. 30, no. 3, pp. 497–508, 2012.

[3] K. Lee, J. Lee, Y. Yi, I. Rhee, and S. Chong, "Mobile data offloading: How much can WiFi deliver?," in *Proceedings of the 6th International Conference*, Co-NEXT '10, (New York, NY, USA), pp. 26:1–26:12, ACM, 2010.

[4] B. Han, P. Hui, V. A. Kumar, M. V. Marathe, J. Shao, and A. Srinivasan, "Mobile data offloading through opportunistic communications and social participation," *Mobile Computing, IEEE Transactions on*, vol. 11, no. 5, pp. 821–834, 2012.

[5] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy consumption in mobile phones: a measurement study and implications for network applications," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, pp. 280–293, ACM, 2009.

[6] F. Mehmeti and T. Spyropoulos, "Is it worth to be patient? Analysis and optimization of delayed mobile data offloading," in *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, pp. 2364–2372, April 2014.

[7] S. Cai, L. Duan, J. Wang, S. Zhou, and R. Zhang, "Incentive mechanism design for delayed WiFi offloading," in *2015 IEEE International Conference on Communications (ICC)*, pp. 3388–3393, June 2015.

[8] M. H. Cheung and J. Huang, "Optimal delayed wi-fi offloading," in *Modeling & Optimization in Mobile, Ad Hoc & Wireless Networks (WiOpt), 2013 11th International Symposium on*, pp. 564–571, IEEE, 2013.

[9] H. Deng and I. H. Hou, "Online scheduling for delayed mobile offloading," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, pp. 1867–1875, April 2015.

[10] H. Deng and I. H. Hou, "On the capacity-performance trade-off of online policy in delayed mobile offloading," *IEEE Transactions on Wireless Communications*, vol. 16, pp. 526–537, Jan 2017.

[11] R. Gass and C. Diot, *An Experimental Performance Comparison of 3G and Wi-Fi", bookTitle="Passive and Active Measurement: 11th International Conference, PAM 2010, Zurich, Switzerland, April 7-9, 2010. Proceedings*, pp. 71–80. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.

[12] A. Balasubramanian, R. Mahajan, and A. Venkataramani, "Augmenting mobile 3G using WiFi," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pp. 209–222, ACM, 2010.

[13] V. F. Mota, D. F. Macedo, Y. Ghamri-Doudane, and J. M. S. Nogueira, "On the feasibility of WiFi offloading in urban areas: The Paris case study," in *Wireless Days (WD), 2013 IFIP*, pp. 1–6, IEEE, 2013.

[14] S. Dimatteo, P. Hui, B. Han, and V. O. K. Li, "Cellular traffic offloading through WiFi networks," in *2011 IEEE Eighth International Conference on Mobile Ad-Hoc and Sensor Systems*, pp. 192–201, Oct 2011.

[15] I. Trestian, S. Ranjan, A. Kuzmanovic, and A. Nucci, "Taming the mobile data deluge with drop zones," *IEEE/ACM Transactions on Networking*, vol. 20,

pp. 1010–1023, Aug 2012.

[16] S. Ha, S. Sen, C. Joe-Wong, Y. Im, and M. Chiang, "Tube: Time-dependent pricing for mobile data," in *Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '12, (New York, NY, USA), pp. 247–258, ACM, 2012.

[17] S. Sen, C. Joe-Wong, S. Ha, J. Bawa, and M. Chiang, "When the price is right: Enabling time-dependent pricing of broadband data," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, (New York, NY, USA), pp. 2477–2486, ACM, 2013.

[18] J. Lee, Y. Yi, S. Chong, and Y. Jin, "Economics of WiFi offloading: Trading delay for cellular capacity," *IEEE Transactions on Wireless Communications*, vol. 13, pp. 1540–1554, March 2014.

[19] I. Rhee, M. Shin, S. Hong, K. Lee, and S. Chong, "Human mobility patterns and their impact on routing in Human-Driven mobile networks," Nov. 2007.

[20] Y.-B. Lin, C.-C. Huang-Fu, and N. Alrajeh, "Predicting human movement based on telecom's handoff in mobile networks," *Mobile Computing, IEEE Transactions on*, vol. 12, no. 6, pp. 1236–1241, 2013.

[21] A. J. Nicholson and B. D. Noble, "Breadcrumbs: forecasting mobile connectivity," in *Proceedings of the 14th ACM international conference on Mobile computing and networking*, pp. 46–57, ACM, 2008.

[22] Y. Li, G. Su, P. Hui, D. Jin, L. Su, and L. Zeng, "Multiple mobile data offloading through delay tolerant networks," in *Proceedings of the 6th ACM workshop on Challenged networks*, pp. 43–48, ACM, 2011.

[23] J. Whitbeck, M. Amorim, Y. Lopez, J. Leguay, and V. Conan, "Relieving the wireless infrastructure: When opportunistic networks meet guaranteed delays," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2011 IEEE International Symposium on a*, pp. 1–10, IEEE, 2011.

[24] X. Hou, P. Deshpande, and S. R. Das, "Moving bits from 3G to metro-scale WiFi for vehicular network access: An integrated transport layer solution," in *Network Protocols (ICNP), 2011 19th IEEE International Conference on*, pp. 353–362, IEEE, 2011.

[25] A. Barbieri, P. Gaal, S. Geirhofer, T. Ji, D. Malladi, Y. Wei, and F. Xue, "Coordinated downlink multi-point communications in heterogeneous cellular networks," in *Information Theory and Applications Workshop (ITA), 2012*, pp. 7–16, IEEE, 2012.

[26] M. Bennis, M. Simsek, A. Czylwik, W. Saad, S. Valentin, and M. Debbah, "When cellular meets WiFi in wireless small cell networks," *Communications Magazine, IEEE*, vol. 51, no. 6, 2013.

[27] S. Singh, H. S. Dhillon, and J. G. Andrews, "Offloading in heterogeneous networks: Modeling, analysis, and design insights," *Wireless Communications, IEEE Transactions on*, vol. 12, no. 5, pp. 2484–2497, 2013.

[28] O. Bilgir Yetim and M. Martonosi, "Adaptive usage of cellular and WiFi bandwidth: an optimal scheduling formulation," in *Proceedings of the seventh ACM international workshop on Challenged networks*, pp. 69–72, ACM, 2012.

[29] N. Buchbinder and J. (Seffi) Naor, "The design of competitive online algorithms via a primal: Dual approach," *Found. Trends Theor. Comput. Sci.*, vol. 3, pp. 93–263, Feb. 2009.

[30] H. Deng and I. H. Hou, "Online job allocation with hard allocation ratio requirement," in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, pp. 1–9, April 2016.

[31] H. Yu, D. Zheng, B. Y. Zhao, and W. Zheng, "Understanding user behavior in large-scale video-on-demand systems," in *Proceedings of the 1st ACM SIGOPS/EuroSys European Conference on Computer Systems 2006*, EuroSys '06, (New York, NY, USA), pp. 333–344, ACM, 2006.

[32] S. Acharya and B. Smith, "Characterizing user access to videos on the world wide web," in *In Proceedings of MMCN*, 2000.

[33] C. Huang, J. Li, and K. W. Ross, "Can internet video-on-demand be profitable," in *in Proceedings of ACM SIGCOMM*, 2007.

[34] X. Cheng, C. Dale, and J. Liu, "Statistics and social network of youtube videos," in *Quality of Service, 2008. IWQoS 2008. 16th International Workshop on*, pp. 229–238, June 2008.

[35] G. Chatzopoulou, C. Sheng, and M. Faloutsos, "A first step towards understanding popularity in youtube," in *INFOCOM IEEE Conference on Computer Communications Workshops , 2010*, pp. 1–6, March 2010.

[36] R. M. Karp, U. V. Vazirani, and V. V. Vazirani, "An optimal algorithm for online bipartite matching," in *Proceedings of the Twenty-second Annual ACM Symposium on Theory of Computing*, STOC '90, (New York, NY, USA), pp. 352–358, ACM, 1990.

[37] X. Cheng, C. Dale, and J. Liu, "Statistics and social network of youtube videos," in *Quality of Service, 2008. IWQoS 2008. 16th International Workshop on*, pp. 229–238, June 2008.

[38] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon, "Analyzing the video popularity characteristics of large-scale user generated content systems," *Networking, IEEE/ACM Transactions on*, vol. 17, pp. 1357–1370, Oct 2009.

[39] N. Littlestone and M. K. Warmuth, "The weighted majority algorithm," 1992.

[40] A. DeSantis, G. Markowsky, and M. Wegman, "Learning probabilistic prediction functions," in *Foundations of Computer Science, 1988., 29th Annual Symposium on*, pp. 110–119, Oct 1988.

[41] N. Littlestone, "Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm," *Mach. Learn.*, vol. 2, pp. 285–318, Apr. 1988.

[42] N. Littlestone, "Redundant noisy attributes, attribute errors, and linear-threshold learning using winnow," in *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, COLT '91, (San Francisco, CA, USA), pp. 147–156, Morgan Kaufmann Publishers Inc., 1991.

[43] R. Armstrong, D. Freitag, T. Joachims, and T. Mitchell, "Webwatcher: A learning apprentice for the world wide web," pp. 6–12, AAAI Press, 1995.

[44] N. Cesa-Bianchi and G. Lugosi, *Prediction, Learning, and Games*. New York, NY, USA: Cambridge University Press, 2006.

[45] M. Hutter, "On the foundations of universal sequence prediction," *CoRR*, vol. abs/cs/0605009, 2006.

[46] G. Goel and A. Mehta, "Online budgeted matching in random input models with applications to adwords," in *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '08, (Philadelphia, PA, USA), pp. 982–991, Society for Industrial and Applied Mathematics, 2008.

[47] C. Karande, A. Mehta, and P. Tripathi, "Online bipartite matching with unknown distributions," in *In STOC*, 2011.

[48] J. Feldman, A. Mehta, V. S. Mirrokni, and S. Muthukrishnan, "Online stochastic matching: Beating 1-1/e," *CoRR*, vol. abs/0905.4100, 2009.

[49] B. Kalyanasundaram and K. R. Pruhs, "An optimal deterministic algorithm for online b-matching," *Theoretical Computer Science*, vol. 233, p. 2000, 2000.

[50] A. Mehta, A. Saberi, U. Vazirani, and V. Vazirani, "Adwords and generalized on-line matching," in *Foundations of Computer Science, 2005. FOCS 2005. 46th Annual IEEE Symposium on*, pp. 264–273, Oct 2005.

[51] S. Moharir and S. Sanghavi, "Online load balancing and correlated randomness," in *Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on*, pp. 746–753, Oct 2012.

[52] D. Bertsimas and J. N. Tsitsiklis, *Introduction to linear optimization*. Athena Scientific series in optimization and neural computation, Belmont, Mass. : Athena Scientific, c1997., 1997.

[53] S. Moharir, S. Sanghavi, and S. Shakkottai, "Online load balancing under graph constraints," *IEEE/ACM Transactions on Networking*, vol. 24, pp. 1690–1703, June 2016.

[54] Z. Mao, C. E. Koksal, and N. B. Shroff, "Optimal online scheduling with arbitrary hard deadlines in multihop communication networks," *IEEE/ACM Transactions on Networking*, vol. 24, pp. 177–189, Feb 2016.

[55] H. Deng and I.-H. Hou, "On the capacity requirement for arbitrary end-to-end deadline and reliability guarantees in multi-hop networks," in *Proceedings of the 2017 ACM SIGMETRICS / International Conference on Measurement and*

Modeling of Computer Systems, SIGMETRICS '17 Abstracts, (New York, NY, USA), pp. 15–16, ACM, 2017.

[56] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *J. ACM*, vol. 20, pp. 46–61, Jan. 1973.

[57] S. Baruah, G. Koren, D. Mao, B. Mishra, A. Raghunathan, L. Rosier, D. Shasha, and F. Wang, "On the competitiveness of on-line real-time task scheduling," in *Real-Time Systems Symposium, 1991. Proceedings., Twelfth*, pp. 106–115, Dec 1991.

[58] S. A. Goldman, J. Parwatikar, and S. Suri, "Online scheduling with hard deadlines," *Journal of Algorithms*, vol. 34, no. 2, pp. 370 – 389, 2000.

[59] M. H. Goldwasser and B. Kerbikov, "Admission control with immediate notification," *J. of Scheduling*, vol. 6, pp. 269–285, May 2003.

[60] M. H. Goldwasser, "Patience is a virtue: The effect of slack on competitiveness for admission control," *Journal of Scheduling*, vol. 6, no. 2, pp. 183–211, 2003.

[61] M. H. Goldwasser and M. Pedigo, "Online nonpreemptive scheduling of equal-length jobs on two identical machines," *ACM Trans. Algorithms*, vol. 5, pp. 2:1–2:18, Dec. 2008.

[62] J. Ding and G. Zhang, *Online Scheduling with Hard Deadlines on Parallel Machines*, pp. 32–42. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006.

[63] J. Ding, T. Ebenlendr, J. Sgall, and G. Zhang, *Online Scheduling of Equal-Length Jobs on Parallel Machines*, pp. 427–438. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.

[64] T. Ebenlendr and J. Sgall, "Approximation and online algorithms," ch. A Lower Bound for Scheduling of Unit Jobs with Immediate Decision on Parallel Machines, pp. 43–52, Berlin, Heidelberg: Springer-Verlag, 2009.

[65] M. Andrews and L. Zhang, "Packet routing with arbitrary end-to-end delay requirements," in *Proceedings of the Thirty-first Annual ACM Symposium on Theory of Computing*, STOC '99, (New York, NY, USA), pp. 557–565, ACM, 1999.

[66] P. P. Bhattacharya, L. Tassiulas, and A. Ephremides, "Optimal scheduling with deadline constraints in tree networks," *IEEE Transactions on Automatic Control*, vol. 42, pp. 1703–1705, Dec 1997.

[67] V. Rodoplu, S. Vadvalkar, A. A. Gohari, and J. J. Shynk, "Empirical modeling and estimation of end-to-end voip delay over mobile multi-hop wireless networks," in *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, pp. 1–6, Dec 2010.

[68] K. Sanada, N. Komuro, and H. Sekiya, "End-to-end throughput and delay analysis for ieee 802.11 string topology multi-hop network using markov-chain model," in *Personal, Indoor, and Mobile Radio Communications (PIMRC), 2015 IEEE 26th Annual International Symposium on*, pp. 1697–1701, Aug 2015.

[69] W. Jiao, M. Sheng, K. S. Lui, and Y. Shi, "End-to-end delay distribution analysis for stochastic admission control in multi-hop wireless networks," *IEEE Transactions on Wireless Communications*, vol. 13, pp. 1308–1320, March 2014.

[70] H. Li, Y. Cheng, C. Zhou, and W. Zhuang, "Minimizing end-to-end delay: A novel routing metric for multi-radio wireless mesh networks," in *INFOCOM*

2009, IEEE, pp. 46–54, April 2009.

[71] R. Li and A. Eryilmaz, "Scheduling for end-to-end deadline-constrained traffic with reliability requirements in multihop networks," *IEEE/ACM Trans. Netw.*, vol. 20, pp. 1649–1662, Oct. 2012.

[72] Q. Wang, P. Fan, D. O. Wu, and K. B. Letaief, "End-to-end delay constrained routing and scheduling for wireless sensor networks," in *2011 IEEE International Conference on Communications (ICC)*, pp. 1–5, June 2011.

[73] I. H. Hou, "Packet scheduling for real-time surveillance in multihop wireless sensor networks with lossy channels," *IEEE Transactions on Wireless Communications*, vol. 14, pp. 1071–1079, Feb 2015.

[74] R. Singh and P. R. Kumar, "Decentralized throughput maximizing policies for deadline-constrained wireless networks," in *2015 54th IEEE Conference on Decision and Control (CDC)*, pp. 3759–3766, Dec 2015.

[75] X. Liu and L. Ying, "Spatial-temporal routing for supporting end-to-end hard deadlines in multi-hop networks," in *2016 Annual Conference on Information Science and Systems (CISS)*, pp. 262–267, March 2016.